VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Pokročilý generátor webových prezentací
Advanced Generator of Web Presentation

2010 Adam Količ

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

Zadání diplomové práce

Student: **Bc. Adam Količ**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Pokročilý generátor webových prezentací

Advanced Generator of Web Presentation

Zásady pro vypracování:

Cílem práce je vytvoření pokročilého generátoru modulárních webových prezentací. Systém umožní vytvářet prezentace složené z různých modulů, tyto modulů dále konfigurovat, upravovat jejich vzhled a případně i upravit jejich funkčnost. Systém bude implementován za pomocí pokročilých technologii typu Ajax.
Práce bude obsahovat:

1. Krátký popis použitých technologii.
2. Analýzu struktury systému.
3. Analýzu ovládacího rozhraní.
4. Popis implementace a vlastní implementaci systému.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Platoš**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Datum odevzdání: 07.05.2010 ……..

Abstrakt

Diplomová práce se zabývá návrhem a realizací systému, pro vytváření a správu modulárních webových prezentací. Zpočátku se věnuje jednoduché analýze a popisu existujících řešení. Dále je provedena analýza a návrh systému a to jak z pohledu programového tak i uživatelského. Je tedy navržena struktura systému a jeho vlastnosti a zároveň je analyzováno ovládací rozhraní systému. Hlavní část práce se pak zabývá samotnou implementací navrženého systému za použití technologií PHP, MySQL či AJAX.

Klíčová slova

generátor, systém, modulární, web, prezentace, PHP, MySQL, AJAX, Smarty

Abstract

Thesis deals with the design and implementation of a system for creating and managing modular websites. The first part devote to simple analysis and description of existing solutions. Next there is made analyze and design of system from the software perspective and user interface. It is therefore proposed structure of the system and its properties, and also analyzed the control interface of system. The main part of the work is then dealt with the implementation of the proposed system by the use of PHP, MySQL and AJAX.

Keywords

generator, system, modular, web, presentation, PHP, MySQL, AJAX, Smarty

Seznam použitých symbolů a zkratek

AJAX - Asynchronous JavaScript and XML, technologie pro asynchronní přenos dat
CMS - Content Management Systém, systém pro správu obsahu
CSS - Cascading Style Sheets, kaskádové styly umožňující popisovat vzhled dokumentu
GNU/GPL - General Public License, všeobecná veřejná licence
HTML - HyperText Markup Language, značkovací jazyk pro tvorbu webových stránek
IDE - Integrated Development Enviroment, vývojové prostředí
MVC - Model-View.Controller, návrhový/architektonický vzor
MySQL - My Structured Query Language, databázový systém
PHP - Hypertext Preprocessor, programovací jazyk
URL - Uniform Resource Locator, specifikuje umístění zdrojů na internetu
W3C - World Wide Web Consorcium, konsorcium stojící za vývojem webových standardů
WWW - World Wide Web, označení pro celosvětovou síť propojených dokumentů
WYSIWYG - What You See Is What You Get, způsob editace dokumentu, při kterém rovnou vidíme výsledek
XAMPP - X, Apache, Mysql, PHP, PERL, multiplatformní program pro snadnou instalaci technologií pro vývoj webových aplikací
XHTML - eXtensible HyperText Markup Language, značkovací jazyk založený na XML
XML - eXtensible Markup Language, rozšiřitelný značkovací jazyk

Obsah

1. Úvod

Na internetu se vyskytuje spousta různých webových aplikací, Frameworků, CMS systémů a dalšího software pro tvorbu webových aplikací či prezentací, ať už pro úplné začátečníky nebo pokročilé programátory a profesionály. Většina těchto aplikací se ovšem zaměřuje právě na určitou skupinu, začátečníky nebo profesionály. Cílem této diplomové práce není tyto systémy kopírovat a snažit se jim nějakým způsobem konkurovat, nýbrž vytvořit takový systém, ve kterém by byli schopni jednoduše pracovat jak úplní začátečníci nedotčení jakýmkoli druhem programování, ale zároveň ponechat pokročilejším uživatelům a programátorům možnost si svůj systém přizpůsobit k obrazu svému.

Práce obsahuje základní popis a seznámení se systémy zabývající se podobnou tématikou, analýzu a návrh vytvářeného systému, návrh rozhraní a samotnou implementaci systému.

2. Popis existujících řešení

V této části se budu zabývat popisem existujících řešení, která se svým oborem týkají této diplomové práce.

Existuje mnoho služeb a programů, které slouží k tvorbě webových prezentací, jejich kvalita a zaměření se však velmi liší. Zhruba by se daly rozdělit do 4 skupin:

- Webové nástroje
- Specializované programy
- Redakční systémy
- Webové Frameworky

2.1 Webové nástroje

Převážně se jedná o služby specializované pro obyčejné uživatele a slouží k vytvoření jednoduchých webových stránek bez potřeby znalosti programování. Kvality a možnosti těchto nástrojů se liší, ale s většinou lze vytvářet jednoduše a velmi rychle poměrně kvalitní prezentace.

2.1.1 WebNode

U nás asi nejznámější nástrojem pro tvorbu webových stránek je internetová služba WebNode, vyvíjená společností Westcom, s.r.o se sídlem v Brně.

Popis

Služba WebNode je navržena zvláště pro laiky a umožňuje tak i běžným uživatelům vytvářet a spravovat vlastní webovou prezentaci. Administrace webu je typu WYSIWYG a veškeré úpravy prezentace, tak probíhají bez jakékoliv znalosti HTML kódu či dalších programovacích technik. Rozhraní je přehledné, do stránky lze vkládat jednotlivé prvky jednoduchým přetažením myši (drag & drop) a na stejné stránce je také editovat.

Systém obsahuje mnoho velmi kvalitně vyvedených grafických stylů. Umožňuje vkládání speciálních gadgetů jako jsou například Google Maps, případně widgetů z knihoven jako jsou Google Gadgets, SpringWidgets a další. Kromě tvorby klasických prezentací a blogů je možné pomocí WebNode vytvořit i jednoduchý elektronický obchod a pomocí speciálního gadgetu jej napojit na platbu pomocí systému PayPal.

Základní verze služby je zdarma. Ta ovšem obsahuje jistá omezení, jako je např. limitovaný prostor na disku pro prezentaci na pouhých 100MB, přenos dat je omezen na 1GB/měsíc, chybí podpora vícejazyčnosti atd. Vyšší placené verze pak již neobsahují taková

omezení a jsou vybaveny více funkcemi. Jejich přesný popis i s výší cen lze najít na domovských stránkách projektu www.webnode.cz.

Výhody a nevýhody

Velkou výhodu toho projektu je velmi kvalitně a přehledně zpracované administrátorské rozhraní, se kterým je schopen vytvořit webové stránky prakticky kdokoliv se základními znalostmi práce s počítačem. Další výhodou je velmi jednoduchá registrace, která požaduje pouze základní údaje, tedy název stránek (část adresy před webnode.cz), e-mail a heslo. Podpora mnoha nejrůznějších gadgetů a widgetů, implementace řady služeb a integrované statistiky pak zkušenějším uživatelům zpříjemní správu vlastní webové prezentace.

Jako nevýhod bych pak viděl příliš striktní omezení u verze nabízené zdarma a pro programátory a kodéry chybějící možnost úprav webu přímo v kódu. K tomu ovšem ani tento systém není určen.

2.1.2 eStránky.cz

Další českou internetovou službou pro snadnou tvorbu webových prezentací jsou eStránky.cz a provozuje ji firma Websitemaster a.s. se sídlem v Praze.

Popis

Služba eStránky.cz je další z internetových služeb zaměřených na běžné uživatele. Ve srovnání s předchozím WebNode však nenabízí tolik možností a přídavných modulů. Není možné vytvořit žádný obsahově složitější web a vše se omezuje pouze na přidávání textového obsahu a fotogalerií. Tato služba je tedy vhodná zvláště pro vytváření jednoduchých osobních stránek nebo blogů.

Podobně jako WebNode i eStránky.cz obsahují řadu grafických stylů a sledování statistik. Zajímavou možností pro pokročilejší uživatele, znalých HTML a kaskádových stylů CSS, je editace kódu jednotlivých částí webových stran.

Další podobnost s WebNode je v cenové politice, kdy základní verze je opět dostupná zdarma s mnoha omezeními a vyšší verze jsou pak za poplatek. Přesné informace jsou opět k nalezení na domovských stránkách projektu www.estranky.cz.

Výhody a nevýhody

Mezi výhody bych zařadil velmi jednoduché rozhraní, se kterým je i běžný uživatel schopen během krátké chvíle vytvořit vkusně vypadající webové stránky.

Nevýhodou pak jsou pak opět striktní omezení pro bezplatnou verzi a nemožnost vytvoření komplexnějších stránek. Ve výsledku pak tedy všechny stránky vytvořené pomocí této služby vypadají velmi podobně.

2.2 Specializované programy

Tyto programy jsou převážně zaměřeny na webdesignéry a kodéry a slouží k vytváření profesionálních webových stránek. Typicky jsou vybaveny WYSIWYG editorem, ale ponechávají možnost úpravy kódu.

2.2.1 Adobe Dreamweaver

Dreamweaver patří ke špičce profesionálních editorů webových stránek. Původně byl vyvíjen společností Macromedia a později odkoupen softwarovým gigantem Adobe, který jej i nadále vyvíjí.

Popis

Na rozdíl od předchozích webových nástrojů pro tvorbu internetových stránek, které sází zejména na jednoduchost a rychlost, je Dreamweaver určen zejména pro pokročilé uživatele, webdesignéry, vývojáře webů a grafické designéry. To ovšem neznamená, že by byl běžný uživatel úplně ztracen a i ten by měl být schopen, díky kvalitnímu WYSIWYG editoru, vytvořit vlastní stránky.

Pomocí tohoto programu je tedy možné vyvíjet a spravovat webové stránky jednoduše pomocí WYSIWYG editoru v podmínkách reálného prohlížeče. Navíc je zde ovšem zachována i možnost úpravy zdrojového kódu. Díky tomu, že v současné době vyvíjí aplikaci firma Adobe, je Dreamweaver provázán s dalšími aplikacemi této společnosti jako jsou Adobe Flash Professional, Adobe Fireworks, Adobe Photoshop a další. V nejnovější verzi program nabízí podporu JavaScriptu s možností rozšíření oblíbených Frameworků například Spry, jQuery nebo Prototype, dále pak vytváření aplikací Adobe AIR, integraci se softwarem Subversion a přímou práci se soubory typu PSD.

V dnešní době dynamických webů, kdy se obsah generuje na straně serverů, je dobré že i DreamWeaver se tomuto trendu přizpůsobil a ve verzi CS4 jsou tak podporovány ASP.NET (C#, VB), Java (JSP), Coldfusion a PHP. Pro tvůrce je tak připravena podpora na straně editoru zdrojového kódu.

Program je distribuován pouze za poplatek, jehož výši lze zjistit na oficiálních stránkách výrobce www.adobe.com a je k dispozici pro operační systémy MAC OS X a Microsoft Windows XP a novější.

Výhody a nevýhody

Mezí výhody tohoto programu patří jeho komplexnost, vzhledem k dlouhodobému vývoji propracované uživatelské rozhraní, kvalitní WYSIWYG editor a provázání s dalšími aplikacemi společnosti Adobe.

K nevýhodám pak patří problémy s diakritikou v módu Live View nebo chybějící český slovník pro kontrolu pravopisu.

2.2.2 Microsoft Expression Web

Microsoft Expression Web je další WYSIWYG HTML editor a nástroj pro správu webových stránek tentokrát od společnosti Microsoft. Tento program je součástí rodiny aplikací Expression, do které dále patří Expression Blend (grafický nástroj pro vytváření grafických interaktivních 3D návrhů), Expression Design (další grafický nástroj kombinující výhody vektorové a bitmapové grafiky) a Expression Encoder (program pro kódování a úpravu video souborů a jejich přípravu pro live streamování).

Popis

Expression Web, který nahradil zastaralý a nepříliš povedený program Microsoft FrontPage, je podobně jako výše uvedený Adobe Dreamweaver profesionální editor pro tvorbu webových prezentací. Založený je na nejpoužívanějších standardech XHTML a CSS a dále podporuje skriptování pomocí JavaScriptu a serverové technologie PHP či ASP.NET. WYSIWYG editor Expression Web ovšem nedosahuje takových kvalit jako editor Adobe Dreamweaver a je tak třeba počítat s tím, že celkový výsledek může v reálu vypadat poněkud jinak. Naštěstí se s programem dodává samostatná aplikace Expression Web SuperPreview, která slouží k testování reálného vzhledu vytvořené webové aplikace v různých prohlížečích. Momentálně jsou podporovány prohlížeče Internet Explorer ve verzích 6, 7 a 8 a Mozilla Firefox ve verzi 3.5. Podpora dalších prohlížečů je v plánu.

Další zajímavou funkcí je Dynamic Web Template, která slouží pro statické webové prezentace, kde se určité části opakují, typicky hlavička, menu nebo patička. Tyto opakující se části je možné nadefinovat staticky a do nich vložit Editable Regions (editovatelnou oblast). Tím budou mít tyto části svůj vlastní obsah a Expression Web se pak postará o zkompletování.

Stejně jako Adobe produkt je i tento distribuován za poplatek s možností vyzkoušení trial verze. Konkrétní informace opět na stránkách výrobce www.microsoft.com.

Výhody a nevýhody

Výhodou je dodržování W3C standardů, možnost validace na přístupnost a kompatibilitu a na rozdíl od Adobe Dreamweaveru dostupná kontrola pravopisu v českém jazyce.

Nevýhodou je pak méně kvalitní WYSIWYG editor oproti tomu v produktu od Adobe a občasná loudavost aplikace Expression Web SuperPreview.

2.3 Redakční systémy

Redakční systémy jsou vlastně předpřipravené webové prezentace či systémy. Většinu je možné volně stáhnout na internetu a nainstalovat na server. Některé Webhostingy dokonce tyto systémy přímo nabízejí ve svém hostingovém programu. Kvalitnější redakční systémy nabízejí širokou podporu šablon, stylů či doplňujících pluginů, které stačí jednoduše doinstalovat.

2.3.1 Joomla

Joomla je open source redakční systém (CMS) postavený na jazyce PHP a databázi MySQL a je dostupný zdarma.

Popis

První verze systému Joomla vyšla v roce 2005 a vycházela z produktu Mambo. Od té doby je neustále vyvíjena skupinou vývojářů sdružovaných na stránkách joomla.org a je vydávána pod svobodnou licencí GNU/GPL.

Systém již v základu nabízí většinu potřebných komponent pro vytvoření zajímavého webu, jako je správa článků, možnost přidávání anket a bannerů a další. Je lokalizován do mnoha jazyků včetně češtiny a jeho funkčnost je možné rozšířit přidáváním rozšíření, kterých za dobu existence Joomly vznikly spousty. Dále jsou podporovány uživatelské účty, či možnost změny vzhledu celého webu pomocí šablon.

Vzhledem k tomu, že je Joomla distribuována zdarma, tak k jejímu používání stačí vlastnit vhodný webhosting a znát potřebné údaje které se zadávají při instalaci, jako jsou například přístupové údaje k databázi.

Výhody a nevýhody

Jednoznačnou výhodou je obrovský výběr dostupných rozšíření a vzhledových šablon. Dále potom poměrně přehledná administrace a lokalizace.

Nevýhodou potom jsou občasné problémy při jiném nastavení serveru, než vyžaduje Joomla. Například problémy s právy k zápisu do adresářů a tím nemožnost jednoduše doinstalovávat nová rozšíření.

2.3.2 Drupal

Drupal je dalším zdarma dostupným redakčním systémem. Taktéž je implementován v jazyce PHP s podporou databázi MySQL a PostgreSQL (v plánu je i podpora pro Oracle MSSQL a další databáze). Pro zajímavost redakční systém Drupal používá úřad prezidenta Spojených států amerických pro své oficiální stránky whitehouse.gov.

Popis

Stejně jako předchozí Joomla je i Drupal vyvíjen pod licencí GNU/GPL. Staví na modularitě, což znamená, že obsahuje malé robustní jádro a na něj jsou napojeny moduly, díky kterým je možné vytvořit nejen obyčejnou webovou prezentaci, ale například i blog nebo e–shop.

I Drupal, vzhledem ke své délce vývoje (vznikl již v roce 2001), disponuje obrovským množstvím modulů, které je možné stáhnout například z oficiálních stránek drupal.org. Také obsahuje správu uživatelů, důležité moduly v základu a nechybí ani lokalizace do češtiny či podpora témat vzhledu.

Výhody a nevýhody

Výhodou je kvalitní jádro a modulární systém s velkým výběrem dostupných modulů a lokalizace.

Nevýhody pak méně přehledná administrace nebo problémy s instalací u některých méně kvalitních webhostingů.

2.4 Webové Frameworky

Framework je jakási nadstavba nad programovacím jazykem sloužící k usnadnění vývoje aplikací. Cílem webových Frameworků je oprostit programátora od nutnosti programování na nízké úrovni a umožnit mu tak zaměřit se na řešený problém.

2.4.1 PRADO

PRADO je webový Framework založený na PHP5, který vytvořil v roce 2004 Qiang Xue a jako inspiraci si vzal Borland Delphi a ASP.NET.

Popis

Jelikož se PRADO inspiruje v ASP.NET tak tento Framework nevychází z klasického návrhového/architektonického vzoru MVC, nýbrž využívá komponent a řízení událostmi, což znamená, že při nějaké akci na webové stránce, například stisknutí tlačítka, se vygeneruje událost, kterou je následně nutné obsloužit.

Mezi základní vlastnosti se řadí oddělená prezentace od logiky, silná podpora databází (MySQL, PostgreSQL, SQLite, MSSQL, Oracle) s několika způsoby přístupu, podpora pro AJAX, internacionalizace a lokalizace, validní XHTML výstup, přátelské URL, cahování, přizpůsobitelný odchyt výjimek a chyb, XML konfigurace, podpora pro validátory a další.

Je to velice povedený Framework volně ke stažení pod licencí revised BSD ze stránek pradosoft.com.

Výhody a nevýhody

Výhodou i nevýhodou je koncepce založená na událostech a záleží na programátorovi, jak mu vyhovuje. Velmi kvalitně je pak řešen přístup k databázím a při správném vývoji pak není problém přejít z jedné databáze na jinou.

Nevýhodou je pak zpočátku orientace v kódu u programátorů, kteří s daným konceptem nemají žádné zkušenosti.

2.4.2 Nette

Nette je dalším zástupcem webových Frameworků. Taktéž je založen na PHP5 ovšem jeho základní koncept vychází z návrhového/architektonického vzoru MVC. Je to český produkt vytvořený Davidem Grudlem.

Popis

Jak už jsem uvedl je Nette založeno na konceptu MVC, což znamená, že vytvářená aplikace je rozdělena na tři nezávislé části. Model, který reprezentuje datovou vrstvu a poskytuje aplikace příslušná data. View (pohled), který převádí data z modelu do formy reprezentovatelné uživateli. A Controller (řadič), který řídí akce od uživatele a provádí příslušné změny v Modelu a View (pohledu).

Mezi základní vlastnosti patří silná podpora zabezpečení (XSS, CSRF a další), integrované ladící nástroje, podpora AJAXu, pěkných URL a dalších moderních technologií. Nette dále nabízí rozšiřitelnost pomocí doplňků, pluginů a komponent a je tedy možné si ušetřit práci zbytečným programováním standardních věcí.

Nette je taktéž šířeno zdarma pod dvěma svobodnými licencemi, Nette Licence a GPL Licence s nimiž je možné Framework používat i pro komerční účely. Veškeré informace je možné získat na oficiálních stránkách projektu nettephp.com.

Výhody a nevýhody

Mezi výhody bych zařadil rozšiřitelnost pomocí pluginů, výkon (rychlost) Frameworku a pro české programátory velkou aktivní komunitu.

Jako nevýhodu bych viděl chybějící podporu databází přímo ve Frameworku (je nutné použít databázovou vrstvu, například dibi).

3. Technologie

V této části bych rád popsal, jakých prostředků jsem k vývoji použil a na čem je celý systém závislý z hlediska svého běhu.

Jako hlavní programovací jazyk pro základ celého systému jsem použil PHP, v době počátku tvorby sytému ve své nejnovější verzi (5.3). PHP jsem si vybral hlavně z důvodu jeho velké rozšířenosti a nezávislosti na konkrétní platformu. Dále by se nešlo obejít bez databázového serveru a k tomu jsem zvolil MySQL z obdobného důvodu jako PHP.

Samozřejmostí u vývoje webových systému je použití XHTML a CSS a ani tento systém nebude výjimkou. Dalšími použitými technologiemi je AJAX a JavaScript hlavně z důvodu dobré uživatelské přívětivosti nejen pro začínající uživatele. Dále je k dispozici šablonovací systém Smarty pro lepší vytváření vzhledu daných modulů v systému a jeho oddělení od aplikační logiky.

Vývoj systému byl veden v systému Microsoft Windows 7 za pomocí vývojového prostředí NetBeans IDE, debugovacího nástroje pro PHP xDebug, a běhového prostředí XAMPP, které obsahuje základní komponenty pro vývoj webových aplikací za pomocí PHP a MySQL.

3.1 XHTML

Je rozšiřitelný značkovací jazyk sloužící pro tvorbu hypertextových dokumentů. Vychází ze staršího jazyka HTML, přebírá ovšem striktnější pravidla definované jazykem XML, jako například názvy elementů a atributů malými písmeny nebo zákaz používání atributů bez hodnoty. Poslední verzí XHTML je 1.1, v plánu byl vývoj verze 2.0, ale ten byl v roce 2009 zastaven a do budoucna se připravuje návrh standardu HTML5.

3.2 CSS

Je jazyk pro definici kaskádových stylů definující vzhled dokumentů popsaných v (X)HTML. Hlavním cílem kaskádových stylů je oddělení obsahu webu od jeho vzhledu do samostatných částí. Původně se i vzhled dokumentů definoval pomocí HTML, ale časem se přišlo na to, že je to slepá cesta. CSS standardizuje konsorcium W3C a v současné době se využívá 2 verze tohoto jazyka, v plánu je pak CSS Level 3.

Ideální kaskádové styly (CSS) se snaží dosáhnout úplného oddělení obsahu od formy (způsob prezentace). Jinými slovy, tvoříte, spravujete a skladujete jádro obsahu webových stránek, webového serveru či aplikace (text, související obrázky, formuláře a tak dále) odděleně od jeho vizuální prezentace (jako je layout, typografie, dekorativní obrázky a podobně). (1)

3.3 JavaScript

Je multiplatformní skriptovací jazyk, který se používá při tvorbě webových stránek. O jeho vykonávání se starají webové prohlížeče a je tedy vhodný pro tvorbu vhodného uživatelské prostředí. V dnešní době je JavaScript využíván v mnoha webových aplikacích od jednoduchých úkonů typu kontroly validace vstupu od uživatele, až po velké robustní systémy, které se pak zdánlivě chovají jako klasické desktopové aplikace, k tomu se ještě ovšem používá následující technologie.

3.4 AJAX

Je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. Tato technologie využívá asynchronního přenosu dat na pozadí. Po vykonání určité akce se pomocí JavaSriptu na server odešle požadavek, který se zpracuje a zpět se pošle výsledek, který je opět pomocí JavaScriptu zpracován.

Hlavní důraz se tedy klade na to, aby online aplikace vypadaly jako desktopové aplikace, což je oblast, ve které AJAX exceluje. Hlavní podíl na této funkčnosti má JavaScript. (2)

3.5 PHP

Je multiplatformní skriptovací interpretovaný jazyk sloužící k vývoji dynamických webových aplikací. Díky tomu, že je tento jazyk interpretovaný, není nutné jej nijak překládat a vytvořený kód se tedy rovnou provádí.

Je to velmi oblíbený jazyk pro tvorbu webových aplikací a jeho obliba by se dala shrnout těmito čtyřmi body. **Upotřebitelnost** – minimalistický jazyk, a to ve dvojím smyslu, v tom co se požaduje od uživatele, i v tom jaké jsou požadavky na syntaxi jazyka. **Vyspělost** – PHP neustále prochází řadou změn a udržuje tak trend moderního programovacího jazyka. **Možnosti** – vývojáři PHP jsou jen výjimečně svázáni jediným možným implementačním řešením. **Cena** – Od svého založení je PHP bez jakýchkoli restrikcí, pokud jde o jeho používání, modifikace a redistribuce. V posledních letech se softwaru s takovouto otevřenou kvalifikací co do licencí říká open-source. (3)

3.6 MySQL

Je databázový server původně vyvinutý švédskou firmou MySQL AB. Nyní jej vlastní společnost Sun Microsystems. Stejně jako PHP je i MySQL velmi populární zvláště ve spojení s tímto programovacím jazykem. Původně bylo MySQL vyvíjeno jako jednoduchý databázový

systém zaměřený především na rychlost, ale časem se doplnily i funkce a vlastnosti z „velkých“ databází a z MySQL se tak stal plnohodnotný systém.

3.7 NetBeans IDE

Je open-source integrované vývojové prostředí pro vývoj softwaru. Toto vývojové prostředí podporuje velkou škálu programovacích jazyků, mezi něž patří Java, C/C++, PHP, JavaScript, Groovy, a Ruby. NetBeans IDE je vytvořeno v jazyce Java a spolu s prostředím Eclipse patří k nejznámějším open-source programům sloužícím k vývoji software. Při vývoji systému jsem použil specializovanou verzi pro PHP, NetBeans IDE for PHP.

3.8 XAMPP

Je open-source multiplatformní program sloužící ke snadné instalaci technologií pro vývoj webových aplikací. Název je složen z použitých technologií a písmeno X znamená, že je multiplatformní. XAMPP tedy obsahuje webový server Apache zajišťující běh webových aplikací, dále databázi MySQL a programovací jazyky PHP a PERL. Kromě tohoto základu ještě XAMPP obsahuje například phpMyAdmin, sloužící pro snadný přístup k MySQL databázi, FTP Server FileZilla a Mercury Mail Transport System.

3.9 Další použité technologie

Kromě výše uvedených hlavních technologií jsem v projektu dále použil tyto specializované nástroje.

3.9.1 Smarty

Smarty je šablonovací systém pro PHP, který umožňuje oddělení aplikační (samotný kód PHP) a zobrazovací (klasické HTML) logiky. To se hodí především pro větší projekty, kdy programátor PHP a kodér HTML není tatáž stejná osoba. Díky Smarty tak kodér úpravami HTML nezničí práci programátora. Kodér tedy vždy pracuje jen s HTML a programátor jen s PHP. Velkou předností je kompilace šablon, kdy není nutné pokaždé šablony překládat do PHP. Ještě mnohem rychlejší než kompilace je cachování obsahu. Cachování je v podstatě kompilace, kdy se nevytváří PHP skripty, ale klasické HTML. (4)

3.9.2 CKEditor

Je textový WYSIWYG editor kompletně napsaný v JavaScriptu. Tento editor slouží k pohodlnému přidávání a editování textu s využitím standardních editovacích možností ve stylu Microsoft Word či OpenOffice Writer. Původně byl tento editor veden pod názvem FCKeditor, ale v roce 2009 došlo k jeho přejmenování. Distribuován je pod svobodnými licencemi GPL, LGPL a MPL nebo také pod komerční licencí CDL.

3.9.3 xDebug

Je rozšíření pro PHP, které slouží k ladění skriptů. Poskytuje mnoho cenných informací a vlastností, mezi které patří například trasování zásobníku a funkcí v chybových výpisech (výpis obsahuje plnou charakteristiku pro uživatelsky definované funkce, název funkce, jméno souboru a řádku označení), alokace paměti, ochrana proti zacyklení, profilovací informace pro PHP skripty, analýzu kódového pokrytí, schopnost interaktivního ladění skriptů pomocí ladícího klienta.

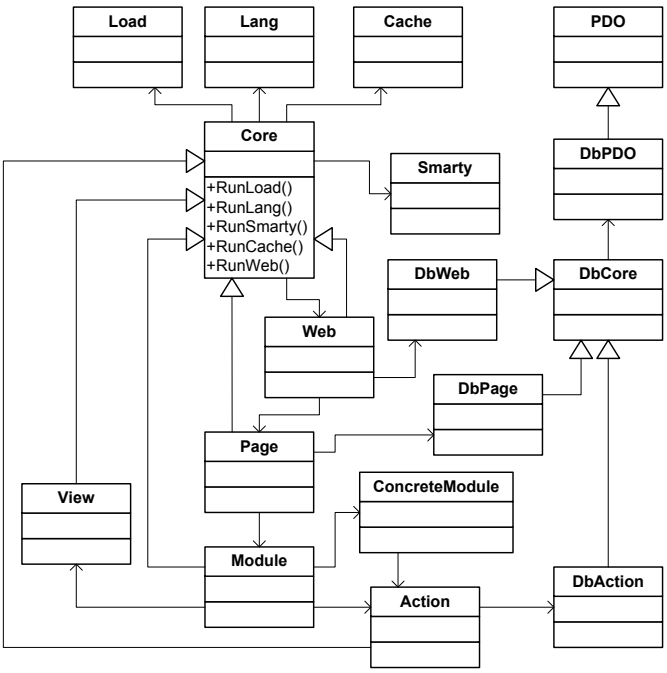
4. Analýza a návrh

Cílem práce bylo vytvořit systém, díky němuž by se daly jednoduše vytvářet a spravovat webové prezentace, ale aby byla zároveň ponechána volnost v úpravách.

Myšlenka je taková, že systém obsahuje základní jádro, které bude dostatečně rychlé, a na něm poběží konkrétní webové prezentace. Čili to znamená, že jádro bude obsluhovat několik aplikací najednou. Každá samotná webová prezentace pak bude obsahovat svou vlastní aplikační část sestávající z tzv. modulů, které budou obstarávat konkrétní funkci webu jako např. správu článku, zobrazení menu a další. Administrace webu bude probíhat ve dvou režimech. V prvním si uživatel vytvářející prezentaci, zvolí jeden z možných vzhledů a rozvržení stránky a doplní ji o potřebné moduly dle svého uvážení. V tomto režimu také půjde dané moduly editovat jak vzhledově, úpravou šablon a stylů, tak funkčně úpravou příslušných skriptů. Ve druhém režimu se pak budou tyto moduly nastavovat či doplňovat o obsah dle funkce konkrétního modulu. Mezi oběma režimy půjde jednoduše kdykoliv přepínat.

4.1 Jádro systému

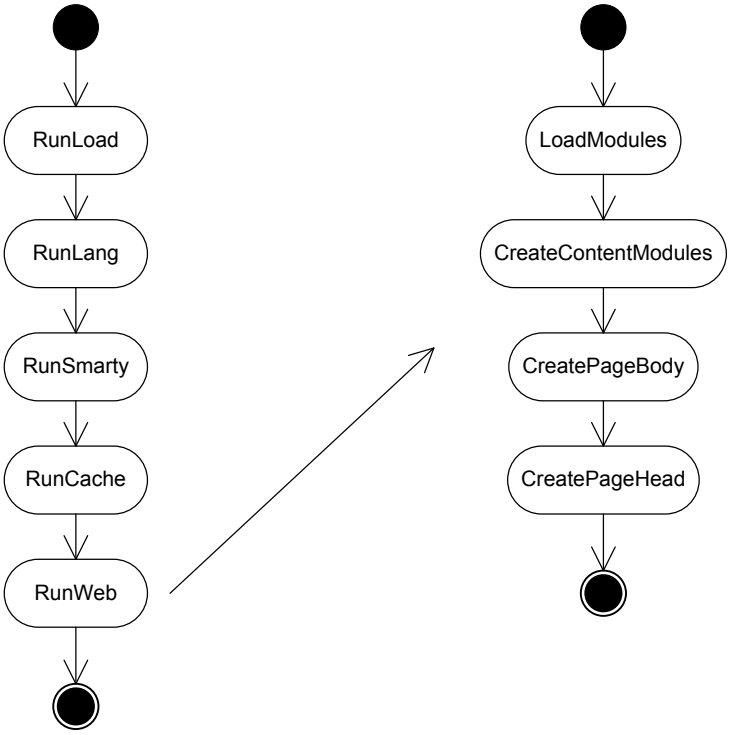
Jelikož se každá webová aplikace skládá ze stránek a v tomto systému se každá stránka skládá z modulů, nabízí se struktura systému sama. Takže velmi jednoduše by se struktura dala popsat takto, jádro se bude skládat z webu, web ze stránek, stránky z modulů a moduly budou sestaveny ze své aplikační části a příslušné šablony (viz obr. 4.1).



4.1 – zjednodušený diagram tříd (jádro)

Na obrázku je vidět rozdělení modulu na konkrétní části a jejich napojení na jádro a administraci. Z obrázku 4.1 je dále patrné odstínění databázové části do tzv. databázového jádra, což umožňuje jednoduchou úpravu přístupu k databázi či kompletní změnu databáze. Takto tedy vypadá návrh ze statického hlediska. K lepšímu pochopení samotné funkce jádra a k přesnějšímu popisu slouží diagram aktivit, jehož obecnou vizualizaci naleznete na obrázku 4.2.

Na počátku se nastaví příslušné cesty k jádru systému a ke klientské aplikaci. Poté systém zjistí, zdali byly odeslány požadavky typu POST nebo GET a provede jejich obsloužení. V dalším kroku se načtou příslušná nastavení a to jak z klientské aplikace, tak i z jádra. Po těchto zaváděcích úkonech se zpracuje volba jazykové mutace a nastartuje se šablonovací stroj Smarty s příslušnými nastaveními. Následuje zavedení cache paměti a spuštění samotného webu.



4.2 – zjednodušený diagram aktivit (jádro)

Část web zajistí zpracování zobrazení webové stránky a spustí vytvoření konkrétní zvolené stránky. Stránka nejprve sestaví své tělo (body) a až poté hlavičku (head) z toho důvodu, aby bylo možné v hlavičce načíst příslušné kaskádové styly a skripty JavaScriptu použité u modulů nacházejících se na dané stránce. Tělo stránky se tedy sestaví tak, že se projde šablona této stránky (případně, pokud existuje vytvořená cache této stránky, se pouze načte tato cache), zjistí se příslušné moduly a jejich šablony a ty se dále sestavují. Moduly se sestaví tak, že se spustí konkrétní aplikační část modulu a načtou se příslušné šablony modulu. Po sestavení

modulu se výsledek předává zpět stránce a po sestavení všech modulů stránka vytvoří obsah hlavičky. Celkový výsledek pak vrátí webu, který se už postará o celkové zobrazení dané stránky.

4.2 Moduly

Jak už bylo napsáno, celá funkčnost webu je postavena na modulech, které se načítají při renderování stránky. Každý modul je nezávislý na jádru a na ostatních modulech se zachováním možnosti spolupráce (komunikace) modulů mezi sebou.

Každý modul se vytváří tak, že se nejprve načtou informace o modulu jako je název, šablona, titulek a popis. A poté se postupně spouští definované metody konkrétního modulu. Nejprve se spustí inicializační metody, které zajistí modulu přístup k aplikaci a při nichž je možné zaslat „zprávu“ ostatním modulům. Dále se volitelně spustí metoda, která se spustí pouze v případě prvního spuštění modulu a může sloužit k potřebným úvodním nastavením, typicky vytvoření tabulek v databázi. Poté se již spustí běh, kdy další akce obstarává část konkrétního modulu a je zde možné přijímat „zprávy“ od ostatních modulů. Po dokončení aplikační části se ke každému modulu připojí jeho definovaná šablona a pomocí Smarty se zpracuje a doplní o požadovaná data. Konečný výsledek se pak předá stránce.

Každý modul dále obsahuje i vlastní část s administrací. Tato část se pak zobrazuje v administračním rozhraní po výběru konkrétního modulu. Více o administraci je popsáno v následující části.

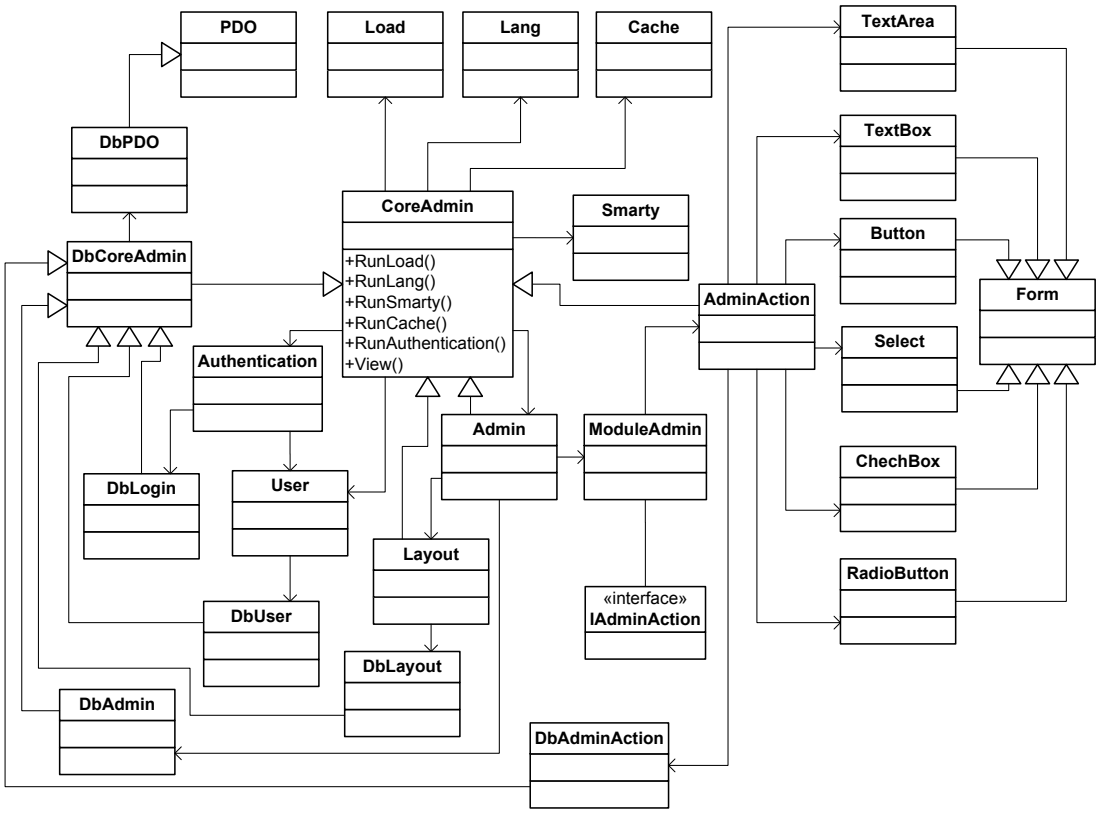
4.3 Administrace

Administrační část je další samostatný úsek systému s vlastním jádrem. To používá pouze při generování výsledného vzhledu modulů v části s editací vzhledu systému. Administrace se tedy dělí na část s editací obsahu modulů a na část s editací vzhledu spolu s možností úpravy kódu modulů.

Řešení rozdělení administrace na tyto dvě části jsem navrhl z toho důvodu, aby nedocházelo ke zmatení uživatele, kdy a jakou část zrovna upravuje, jestli je to pouze obsah nebo mění vzhled atd. Samostatná část pro správu obsahu modulů je tedy jednotná. Všechny moduly se řídí dle speciálního rozhraní, díky kterému je možné udržet stejný vzhled a základní funkčnost správy všech modulů. Administrace vzhledu je pak řešena systémem drag & drop (chyť a pusť), kdy ze speciálního panelu modulů lze tyto moduly libovolně umisťovat na stránku. Vzhled modulu se pak okamžitě generuje po vložení na danou pozici. Podrobnější popis a návrh ovládacího rozhraní se nachází v další části textu.

4.3.1 Jádro

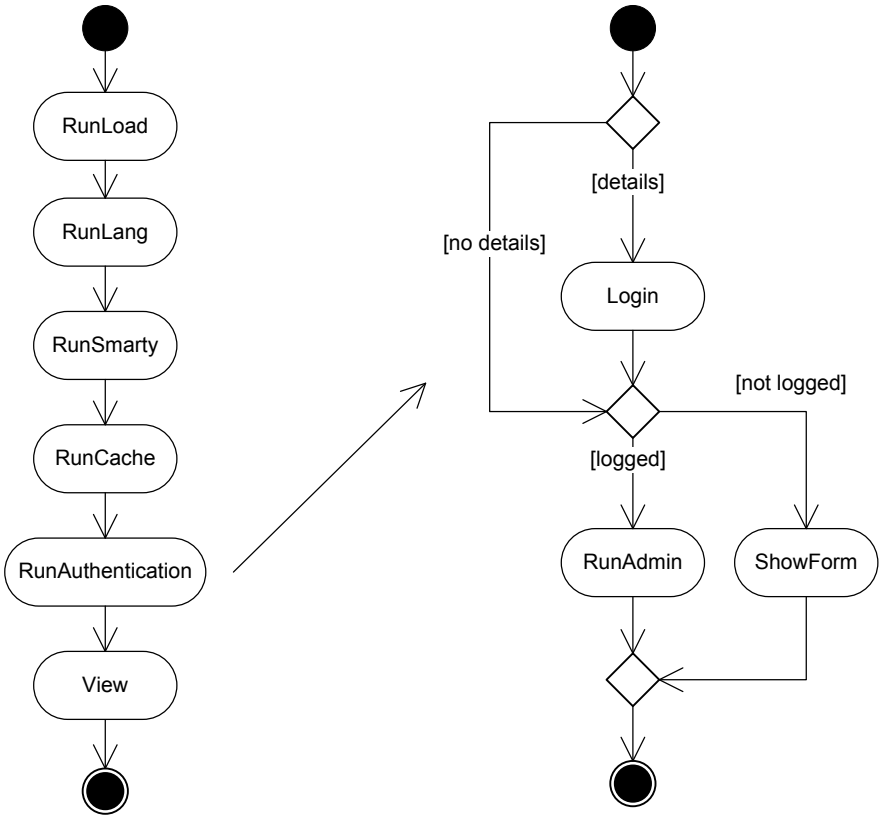
Z obrázku 4.3 je patrné, že základní třídy jsou shodné s jádrem systému, ovšem dále se administrace liší. Je zde přítomna třída zajišťující autentizaci uživatele a třídy umožňující práci s jednotlivými částmi administrace. Třída Layout zajišťuje správu vzhledu a třída ModuleAdmin pak reprezentuje konkrétní administraci vybraného modulu. Dále obsahuje třídy reprezentující jednotlivé části formuláře, jako jsou tlačítka textová pole a další. Nesmí chybět ani připojení k databázi a třída pro práci s uživatelem.



4.3 – zjednodušený diagram tříd (Administrace)

I aplikační část jádra administrace je zpočátku shodná s jádrem aplikace (obr 4.4). Nejprve se tedy načtou potřebné informace a nastavení, připojí se slovník zvolené jazykové mutace, nastaví se a spustí šablonovací stroj Smarty a připraví se práce s cache pamětí. Po těchto nutných úvodních procedurách se spustí autentizace uživatele. Zjistí se, zdali byl vyplněn přihlašovací formulář a pokud ano provede se přihlášení, v opačném případě se přihlášení přeskočí. V dalším kroku se ověří, jestli přihlášení proběhlo správně, resp. jestli je uživatel přihlášen, a podle toho se buď zobrazí přihlašovací formulář, pokud uživatel není přihlášen,

anebo se spustí administrační rozhraní, které dále určuje, co se bude vykonávat (správa modulů, správa vzhledu atd.).



4.4 – zjednodušený diagram aktivit (Administrace)

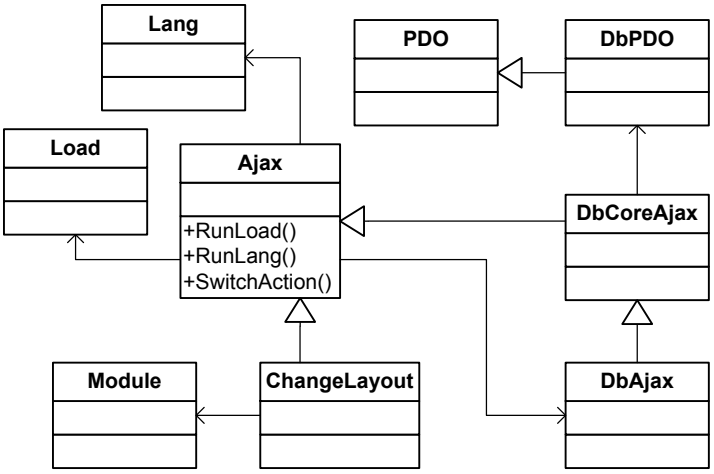
4.3.2 AJAX

AJAX, jak už bylo popsáno v části o technologiích, slouží k asynchronní komunikaci mezi klientem a serverem za použití JavaScriptu a XML. V systému je tedy této technologie využito hlavně v části s administrací vzhledu stránky webové prezentace.

Pomocí asynchronního přenosu je tedy řešeno načtení vzhledu a obsahu daného modulu po jeho vložení, nové načtení vzhledu po změně stylu modulu, uložení rozložení (vzhledu) dané stránky. Dále pak základní editace modulů (změna názvu a popisu) a stránek (nastavení domovské stránky, zákaz zobrazení stránky).

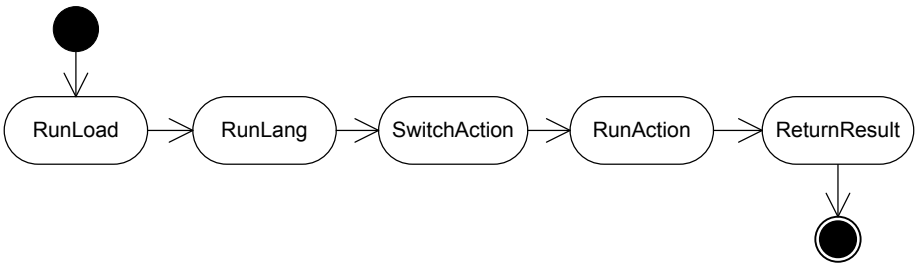
Samostatně je pak řešena realizace editace modulů ve smyslu úpravy jejich vzhledu a funkčnosti. Tato část systému umožňuje úpravu zdrojových kódů vybraného modulu a to přímo v prostředí webového prohlížeče. Pomocí technologie AJAX se tedy „natáhnou“ příslušné kódy do textového pole, kde je může uživatel upravovat a po uložení se opět asynchronně odešlou na server, kde se uloží. Podobně pak funguje i aktualizace modulu v reálném čase, kdy se změna

v kódu okamžitě (celková prodleva závisí na rychlosti připojení a odezvě) projeví na chování modulu. Dále je umožněno vytváření nových souborů, jejich přejmenování a vymazání opět pomocí AJAXu.



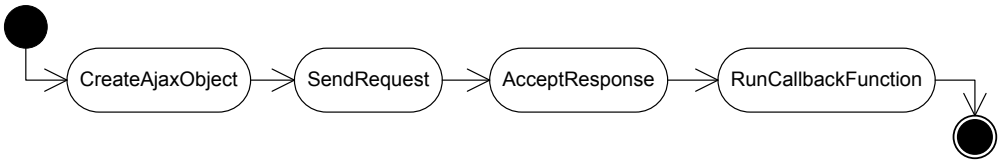
4.5 – zjednodušený diagram tříd (AJAX - server)

Statický diagram (obr. 4.5) serverové části, která obstarává vlastní vykonání požadované funkce je jednodušší než předchozí diagramy a to z jednoduchého důvodu. U této technologie je nutné, aby se výsledek požadavku vrátil klientovi co nejdříve a uživatel tak neměl pocit, že je systém pomalý. Hlavní třída Ajax tedy využívá pouze tříd Load a Lang a vytváří připojení k databázi prostřednictvím třídy DbAjax. O samotné provedení požadovaných funkcí se pak stará třída ChangeLayout, která navíc vytváří instanci třídy Module, pokud se pracuje s moduly stránky.



4.6 – zjednodušený diagram aktivit (AJAX - server)

To jak funguje serverová část, je pak znázorněno na obrázku 4.6. Je tedy patrné, že vše funguje na velice jednoduchém principu, kdy se na začátku načtou základní nastavení a jazykové slovníky, v dalším kroku se provede výběr příslušné akce podle získaného požadavku od klienta a následně se tato akce provede. Po provedení se zpracují výsledky a ty se odešlou zpět klientovi.



4.7 – zjednodušený diagram aktivit (AJAX - klient)

Na straně klienta pak pomocí JavaScriptu vše probíhá takto. Nejprve se vytvoří objekt metody Ajax, v němž se nastaví potřebné údaje, následně se pomocí tohoto objektu odešle požadavek na server, kde se zpracuje (viz předchozí odstavce), a po přijmutí odpovědi se spustí definovaná Callback funkce, která přijme data odeslaná serverem a provede příslušné definované akce.

4.4 Databáze

Systém je navržen tak, aby pracoval se dvěma různými databázemi. Jedna je společná všem klientským aplikacím (jednotlivým prezentacím) a ta zajišťuje správu uživatelů této aplikace. Další databází má pak každá klientská aplikace vlastní a ta obsahuje tabulky stránek, jazykových verzí a hlavně tabulky jednotlivých modulů.

4.4.1 Hlavní databáze

Hlavní databáze je tedy jen jedna a ta spravuje uživatele jednotlivých klientských aplikací. Obsahuje tedy tabulky se seznamem uživatelů (tabulka ‘název’admin, za název se doplní konkrétní název aplikace), tabulku s výběrem rolí (tabulka role) a tabulku s definicí zakázaných modulů pro daný typ role (tabulka disable_role_module). Přesný popis jednotlivých tabulek je uveden v datovém slovníku.

Datový slovník

Název	Typ	Velikost	Klíč	Null	Index	Poznámky
Tabulka: ‘název’admin						
id	int	4	Y	N	Y	
id_role	int	4	N	N	N	ID pro spojení s tabulkou role
nick	varchar	64	N	N	N	Uživatelské jméno
pass	varchar	40	N	N	N	Heslo kryptované pomocí SHA1
Tabulka: role						
id	int	8	Y	N	Y	
type	varchar	64	N	N	N	Typ role (admin, user)
editModules	tinyint	1	N	N	N	Určuje zda je povolena editace kódu modulu
Tabulka: disable_role_module						
id_role	int	8	Y	N	Y	ID pro spojení s tabulkou role
moduleName	varchar	256	Y	N	Y	Název modulu

4.4.2 Klientské databáze

Jak již název napovídá, tak každá klientská aplikace má databázi vlastní. Společné všem těmto databázím (obsah je samozřejmě různý) jsou tabulky se seznamem stránek (tabulky core_pages a core_pages_lang) a tabulka s jazyky (tabulka core_langs). Další tabulky pak patří jednotlivým použitým modulům a jejich struktura i obsah závisí na těchto modulech.

Datový slovník

Název	Typ	Velikost	Klíč	Null	Index	Poznámky
Tabulka: core_langs						
id	int	4	Y	N	Y	
code	varchar	5	N	N	N	Kód jazyka a dialektu (př. cs_cz, en_us)
name	varchar	32	N	N	N	Název v jayka
Tabulka: core_pages						
id	int	8	Y	N	Y	
active	tinyint	1	N	N	N	Určuje zda je stránka aktivní (viditelná)
is_homepage	tinyint	1	N	N	N	Určuje zda je stránka „domovská“
Tabulka: core_pages_lang						
id	int	8	Y	N	Y	
id_pages	int	8	N	N	Y	ID pro spojení s tabulkou core_pages
id_langs	int	4	N	N	Y	ID pro spojení s tabulkou core_langs
title	varchar	256	N	N	N	Titulek stránky
rewrite_title	varchar	64	N	N	N	Přepisovací titulek zobrazený v URL adrese
Struktura dalších tabulek je definována podle konkrétních modulů						

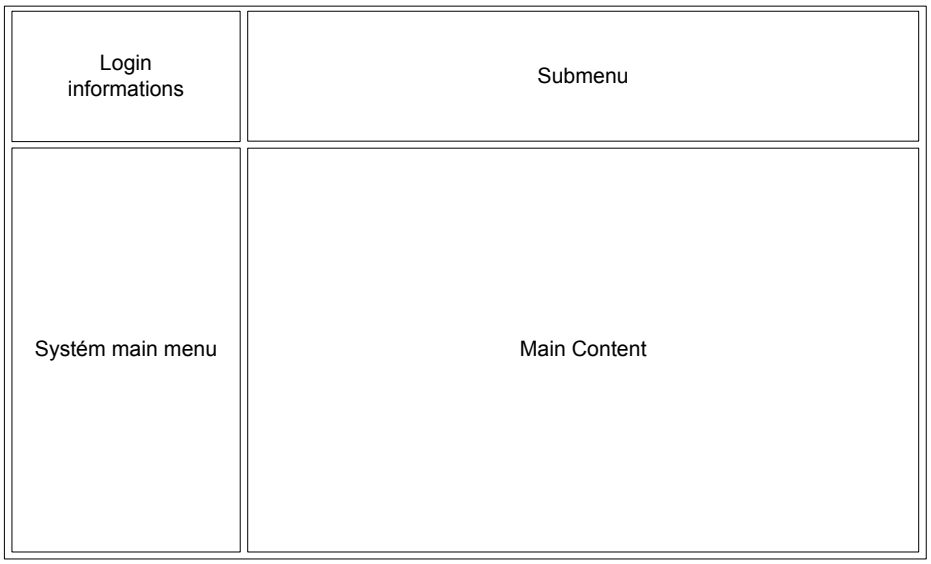
4.5 Návrh ovládacího rozhraní

Při návrhu vzhledu ovládacího rozhraní a jeho funkčnosti jsem se snažil dosáhnout maximální jednoduchosti a přehlednosti. Nakolik se mi to povedlo, již budou muset posoudit samotní uživatelé.

4.5.1 Správa modulů

Na obrázku 4.8 je znázorněno blokové schéma rozvržení administrační části pro správu obsahu modulů. Tato část tedy obsahuje 4 oblasti, z nichž každá představuje vlastní obsah. Oblast vlevo nahoře (Login informations) slouží k zobrazení informací o přihlášeném uživateli a umožňuje mu zvolení jazykové mutace administrace a odhlášení. Druhá oblast vlevo dole (Systém main menu) zobrazuje hlavní menu. Zde se objeví hlavně seznam modulů, u kterých je

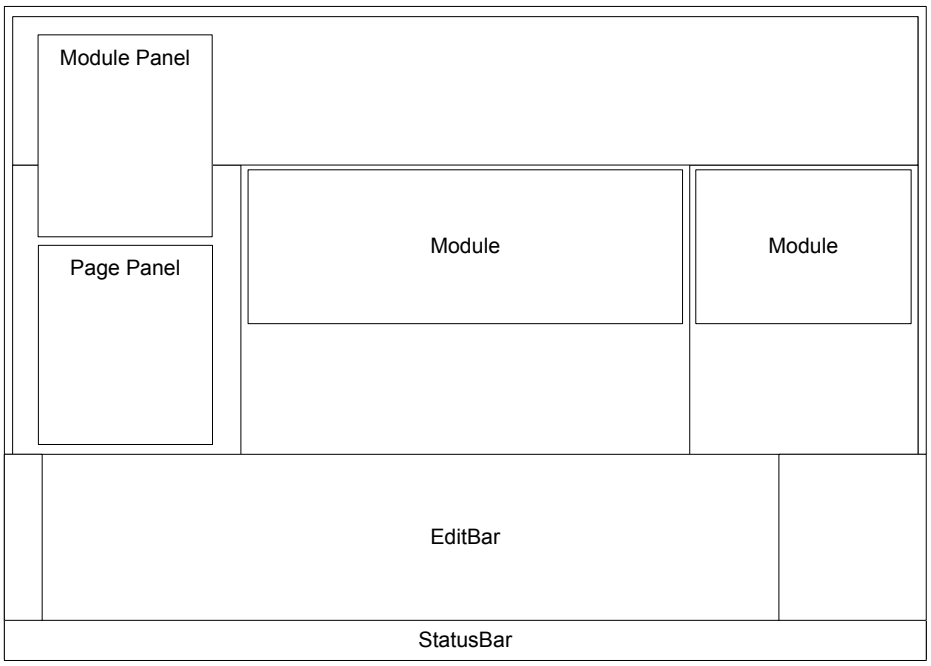
možno editovat obsah. Další oblast vpravo nahoře (Submenu) pak zobrazí menu nižší úrovně daného modulu, pokud je u tohoto modulu obsaženo. Týká se to hlavně složitějších modulů, které mají větší možnosti nastavení. Poslední oblast (Main Content) je největší a ta zobrazuje obsah administrace vlastního modulu. Zde se bude objevovat například tabulka se seznamem položek modulu nebo formulář pro přidávání či editaci těchto položek.



4.8 – blokové schéma (správa modulů)

4.5.2 Správa vzhledu

Na dalším obrázku (obr. 4.9) je návrh části se správou vzhledu. Největší část obrazovky v tomto případě samozřejmě zabírá zobrazení dané stránky se svými moduly, u kterých je možné měnit styl a případně je také editovat. Jednotlivé části stránky jsou ohraničeny podle vybraného rozvržení stránky a to kvůli přehlednému označení oblastí, do kterých je možné vkládat moduly. Dále správa vzhledu obsahuje dva panely. První obsahuje seznam všech použitelných modulů (Module Panel), které je možno chycením a přetáhnutím jednoduše umístit na zvolenou pozici ve stránce. Tento panel navíc umožňuje přidávat nové moduly, mazat stávající a upravovat jejich info (název a popis). Druhý panel (Page Panel) pak obsahuje seznam stránek a umožňuje jejich editaci (název, rozvržení, šablona, zakázání, domovská stránka), vytváření nových a mazání. Oba panely obsahují funkci pro sbalení, kdy zůstane zobrazen jen popisek panelu pro opětovné rozbalení.



4.9 – blokové schéma (správa vzhledu)

Po stisku tlačítka pro editaci modulu se ve spodní části vysune oblast (EditBar) sloužící ke správě daného modulu. Tato oblast zabírá třetinu stránky a je dále rozdělena na pruh tlačítek, které představují základní práci s otevřeným modulem, tedy uložení, vrácení změn, přejmenování, vytvoření nového a smazání stylu modulu. Další, hlavní oblast, poté obsahuje panel se záložkami reprezentující jednotlivé části, ze kterých se skládá daný modul a textové pole s obsahem vybraného souboru. Dále tato část obsahuje tlačítka pro maximalizaci editační oblasti na celou stránku a tlačítko pro zavření editační oblasti. Poslední, třetí část, pak zobrazuje seznam souborů daného typu a jejich základní editaci (nový, přejmenovat, vymazat).

Úplně ve spodní části stránky se pak nachází oblast, která uživatele informuje o prováděné činnosti a obsahuje tlačítka pro návrat do administrace modulů, uložení nového rozvržení (rozmístění modulů) a tlačítko pro zrušení neuložených změn.

5. Implementace

Jak již bylo v předchozích kapitolách zmíněno, hlavním programovacím jazykem je PHP a to ve své nejnovější verzi (5.3). Nejpodstatnější změnou této verze je přidání podpory jmenných prostor, která zajišťuje jednodušší správu aplikace a řeší dřívější problémy u větších aplikací, kdy mohlo docházet ke kolizi názvů tříd, metod apod. Implementovaný systém tedy této nové vlastnosti bohatě využívá. Zvláště v případě mnoha modulů je to velmi vítané, jelikož se v nich vyskytují metody se shodnými názvy a bez podpory jmenných prostor bych to musel řešit pomocí prefixů či sufixů.

Níže uvedený text slovně popisuje implementaci systému. Tento text nemá za cíl konkrétně popsat celý vytvořený kód, ale pouze osvětlit využití tříd a jejich metod. Konkrétní popis každé metody i se svými parametry a návratovými hodnotami je popsán v přiložené programátorské dokumentaci.

5.1 Jádro

5.1.1 Architektura jádra systému

Co se týče architektury, je systém navržen tak, aby byly od sebe odděleny nejen datová, aplikační a prezentační část, ale také aby od jádra byly určitým způsobem odděleny i moduly. Zároveň by však měly být zachovány možnosti využití základních prostředků systému všemi jeho částmi.

Jak je to tedy v praxi. Hlavní třídou jádra je třída Core, která obsahuje, mimo jiné, statické chráněné proměnné dostupné pouze potomkům této třídy. Tyto proměnné obsahují instance základních tříd, které využívá celé jádro a je tedy možné v rozšířených třídách těchto proměnných využívat. V praxi to pak znamená například to, že se smarty šablonami může pracovat nejen část systému, která spravuje stránky (třída Page), ale také část s moduly (třída Module) apod.

Hlavní třídou je tedy Core, která dále vytváří instanci třídy Web, ta vytváří instanci třídy Page, a ta vytváří instanci třídy Module. Tato třída, pak vytváří instance tříd View a Action, které se starají o vzhled a aplikační propojení s konkrétním modulem. Všechny tyto uvedené třídy pak dědí z třídy Core z důvodu uvedeného výše.

Konkrétní modul, resp. instance třídy reprezentující tento modul, která je vytvořena ve třídě Module, je pak postaven bokem a k jádru systému má přístup pouze v rámci vlastní instance třídy Action. Třídy využívající připojení k databázi, konkrétně Web, Page a Action, pak vytvářejí instance svých databázových tříd, které jsou rozšířením databázového jádra.

5.1.2 Popis tříd jádra systému

index.php

Jádro je hlavní část systému, která celou aplikaci sestavuje a pohání. Jako u každé aplikace v PHP (pokud není server nastaven jinak) je implicitním souborem pro spuštění, soubor index.php. Tento soubor obsahuje pouze následující. Metodu zpracovávající odchytávání chyb a výjimek, dále připojuje soubor autoload.php pro zajištění automatického načítání souborů použitých tříd. A nakonec to hlavní, spouští aplikaci vytvořením instance třídy Core.

Core

Třída Core, patřící jako všechny třídy jádra do jmenného prostoru Core, zajišťuje spuštění a běh aplikace. Nepoužívá žádné veřejné proměnné ani metody a je tak vůči vnějšímu zásahu zcela imunní. V konstruktoru přebírá dva povinné a jeden nepovinný parametr. První dva určují cestu k jádru aplikace a cestu ke klientské aplikaci na serveru. Třetí, nepovinný parametr, pak využívá administrace k vygenerování obsahu modulu v části se správou vzhledu. V konstruktoru se pak dále spouští privátní metody, zajišťující načtení a správu nastavení (třída Load), načtení a správu jazykové mutace (třída Lang), nastavení a spuštění šablonovacího stroje Smarty (třída Smarty), spuštění vyrovnávací paměti (třída Cache) a nakonec spuštění generování webu. Kromě těchto metod dále obsahuje chráněnou metodu GetSelectedPage, která vrací aktuálně zobrazovanou stránku, a destruktor, který po skončení odstraní záložní soubory z adresáře tmp v klientské aplikaci.

Load

Load je třída, která se spouští ihned po spuštění aplikace a má za úkol načíst potřebná nastavení a další podpůrné funkce. Nejprve je testováno, zdali byl na server odeslán požadavek typu POST. Pokud ano dojde k jeho uložení do „sezení“ (SESSION) a aplikace je přesměrována na stejnou adresu, na kterou byl požadavek vyslán. Poté se ze sezení zkopíruje obsah předešlého požadavku do privátní statické proměnné a ze sezení se odstraní. Pokud byl zároveň na server nahrán soubor, tak se zkopíruje do adresáře tmp klientské aplikace a tam je k dispozici do ukončení běhu skriptu. Tato funkce je implementována z důvodu odstranění opakovaného odeslání dat po obnovení stránky (například při vkládání nového příspěvku do návštěvní knihy). Data odeslaná metodou POST a nahrané soubory jsou poté dostupné pomocí konkrétních veřejných metod.

Po zpracování požadavku POST je na řadě požadavek GET. Kvůli zajištění tzv. „hezkých URL“ jsou veškeré požadavky odeslané metodou GET ve tvaru „?name=value“, převedeny na tvar „/name/value/“ a jejich hodnoty jsou opět dostupné pomocí veřejné metody třídy Load. Po zpracování požadavků je nutné načíst nastavení jádra i klientské aplikace. Nastavení se nacházejí v adresáři settings a jsou ve tvaru ini souboru. Metoda načítající tato nastavení tedy jen potřebuje cestu k těmto adresářům a o vše ostatní se postará. Přístup

k načteným nastavením je opět možný pomocí veřejných metod, zvlášť pro klientské a zvlášť pro hlavní.

Lang

Po úvodním nastavení a zpracování požadavků je třeba zpracovat jazykové nastavení. Nejprve se nastaví zvolený jazyk, pokud jazyk nebyl uživatelem zvolen, použije se ten, který je zapsán v klientově nastavení. Případně pokud byla zadána URL adresa stránky vztahující se k nějakému konkrétnímu jazyku, je zvolen tento. Poté se načtou slovníky dle zvoleného jazyka. Jedná se o slovníky jádra, moduly můžou a nemusí mít vlastní. Tyto slovníky jsou stejně jako nastavení ve formátu ini souboru. K získání kódu aktivního jazyka a získání obsahu slovníku jsou implementovány speciální veřejné metody.

Smarty

Následuje spuštění třídy Smarty a její základní nastavení. Nastavuje se cesta k adresáři se šablonami (nastaveno na domovský adresář klienta), dále cesta k adresáři, kde se budou ukládat zkompilované šablony pro rychlejší načítání (Templates_c) a nastaví se zabezpečení. Přesněji je zapnuta direktiva security, která například zabraňuje vkládání php kódu přímo do šablony. Konkrétní popis direktivy security, jakož i celou dokumentaci lze nalézt na domovských stránkách projektu www.smarty.net (popisovat konkrétně šablonovací systém Smarty není obsahem této práce).

Cache

Cache slouží k dočasnému ukládání často načítaných informací. V jádru systému se využívá k ukládání použitých modulů na každé stránce, aby nebylo nutné při každém zobrazení stránku procházet a vyhledávat použité moduly. Přeci jen po úvodním nastavení, se moduly tak často na stránce nemění a není je tedy nutné vždy vyhledávat. Po spuštění se tedy nastaví cesta k adresáři, kde se tyto soubory budou ukládat (cache) a také jejich přípona (cch). Dále je možné ještě použití cache zakázat. Třída tedy obsahuje metody pro uložení do paměti, kdy jsou vstupní data serializována do podoby uložitelného textu, a načtení z paměti, kdy jsou data zpětně deserializována a vrácena v původní podobě. Dále je to metoda sloužící ke zjištění zda daný soubor v cache existuje, metody k vymazání cache a to buď kompletně, nebo podle definovaného názvu a metoda, která cache vypne.

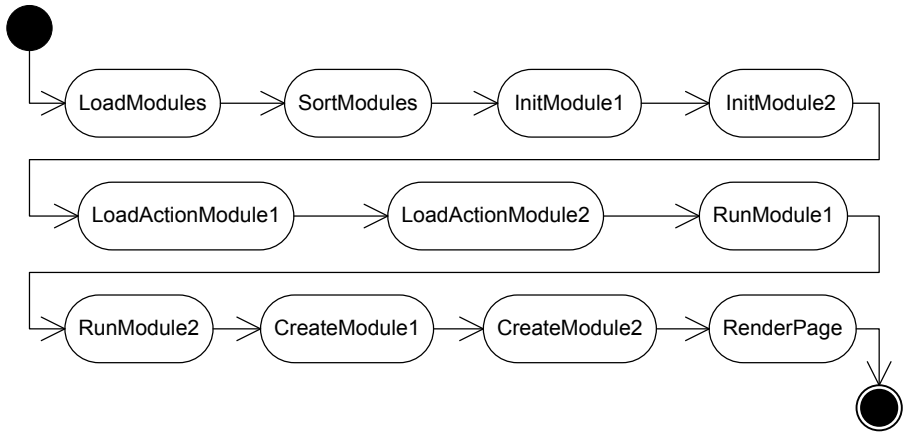
Web

V tomto okamžiku je tedy vše připraveno k sestavení aplikace a zobrazení aktuální stránky. Zjistí se, která stránka je aktuálně zvolena, nastaví se smarty proměnné s aktuální a předchozí URL adresou a spustí se metody třídy Web, které započnou vytváření webové prezentace. Třída Web tedy obsahuje metodu pro vytvoření webu, ve které se nastaví ohraničeni stránky a vytvoří se instance třídy Page, a metodu pro vyrenderování webu, která zobrazí

kompletně vygenerovanou stránku. Zbývající metody potom slouží k nastavení a získání aktivní stránky.

Page

Vytváření samotné stránky pak probíhá takto. Provede se připojení k databázi, ze které se podle přepisovacího titulku zjistí veškeré informace o stránce a následně se načte její šablona. Následně se začne generovat tělo (body) stránky. Algoritmus projde šablonu a vyhledá v ní všechny značky (tagy) identifikující vložený modul. Z těchto značek zjistí název modulu, zvolený styl a závislost na jiném modulu (pokud jsou dané moduly stránky umístěny v cache, tak se šablona neprohledává). Podle těchto zjištěných informací jsou následně vytvořeny instance třídy Module. Pokud má nějaký modul nastavenu závislost na jiném, například z důvodu, kdy je třeba, aby jeden modul počkal na vykonání kódu druhého modulu, algoritmus seřadí tyto moduly tak, že ty závislé se vykonají, až po těch na kterých závisí. Po tomto seřazení jsou spuštěny metody pro vygenerování modulu. V dalším kroku jsou značky identifikující modul nahrazeny obsahem vygenerovaného modulu.



5.1 – příklad vytvoření stránky se dvěma moduly

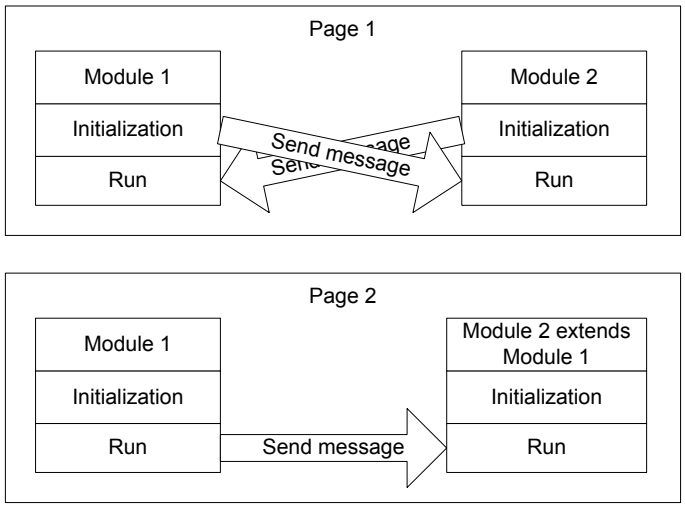
Po výše uvedeném sestavení těla stránky je třeba vygenerovat hlavičku (head). Ta se generuje až jako druhá, jelikož je nutné do ní zahrnout použité kaskádové styly a externí JavaScriptové kódy z jednotlivých modulů. Nejprve se tedy provedou metody, zjišťující použité JavaScriptové kódy a kaskádové styly, které vrací přímo obsah v HTML kódu. Tento kód se do hlavičky vloží a vše se uzavře přidáním titulku. Tím je generování hlavičky ukončeno.

Module

Moduly jsou klíčovou částí vytvářené prezentace a právě ony určují výslednou funkčnost. Vytváření modulu probíhá v několika krocích. Nejprve se vytvoří instance třídy Module a načtou se informace ze souboru info.xml, který by měl každý modul obsahovat. V dalším kroku se vytvoří instance třídy Action a instance konkrétního modulu. Instance třídy

Action se pak tomuto konkrétnímu modulu předá. Následně se spustí inicializační metoda modulu a nastaví se Smarty proměnné definující tento modul a cesty k používaným adresářům (others, images, files). Pak už následuje samotné spuštění modulu a o další běh se stará kód konkrétního modulu. Nakonec se načte šablona vybraného stylu, do které se doplní případná data.

Jak jsem již dříve naznačil, moduly mezi sebou mohou určitým způsobem komunikovat. Slouží k tomu systém „zpráv“, které si mohou během generování moduly zaslat. Tento systém má následující pravidla a omezení. Standardně je možné odeslat „zprávu“ v inicializační části jednoho modulu a přijmout ji v části běhové. Takto lze „zprávy“ posílat mezi moduly libovolně. Pokud je ovšem nutné počkat na vykonání kódu prvního modulu a až poté „zprávu“ předat druhému, musí tento druhý modul rozšiřovat modul první. V opačném směru zprávu zaslat nelze.



5.2 – Blokové schéma možností posílání zpráv mezi moduly

Action

Action je třída, která modulům umožňuje přístup k jádru systému, jelikož jsou moduly jako takové samostatné a do jádra nemají povoleno zasahovat. Zprostředkovává tedy přístup k databázi, umožňuje nastavit a načíst Smarty proměnné využívané v šablonách, odeslat a přečíst zprávy zasílané mezi moduly, dále zprostředkovává přístup k zaslaným požadavkům typu GET i POST, obsahuje metody k získání přehledu stránek a jazyků dané prezentace a metodu k nastavení titulku stránky.

View

Třída, která pracuje se vzhledem modulu. Z názvu modulu a názvu jeho šablony nalezne příslušnou šablonu, doplní ji o data a vrátí výsledný HTML text, který je posléze předán stránce.

Kromě toho ještě načítá CSS styly jednotlivých modulů, které se pak načítají do hlavičky při vytváření stránky.

Konkrétní modul

Konkrétní modul samozřejmě není název třídy. Každý modul má totiž svou vlastní třídu pojmenovanou podle svého názvu a je umístěn ve jmenném prostoru Modules a podprostoru se svým jménem, například \Modules\Guestbook. Díky tomu jsou i jednotlivé moduly mezi sebou odděleny a nemůže tak dojít ke kolizi názvů. Každá tato třída implementuje rozhraní Iaction, které definuje povinné metody. Jsou to metody zajišťující inicializaci, připojení instance třídy Action a metodu spouštějící běh modulu. Další metody či dokonce třídy může vytvořit tvůrce modulu za podmínek, že dodrží jednoduché pravidlo o zachování stejného jmenného prostoru.

Protection

Protection je speciální třída, obsahující jedinou metodu, která zabraňuje v přístupu do jádra z jiných jmenných prostor než je Core a DbCore (jmenný prostor pro třídy pracující s databází).

Special

Tato abstraktní pomocná třída obsahuje veřejné statické metody, které vykonávají univerzálně použitelné funkce. Je zde metoda, která rekurzivně projde zadaný adresář a vrátí jeho kompletní strukturu ve vícerozměrném poli. Dále je to metoda, která vymaže zadaný adresář i s jeho obsahem, a metoda, která zkopíruje zadaný adresář i s jeho obsahem do jiného zadaného adresáře. Jako poslední je to metoda, která na vstupu přijímá obsah CSS souboru, který rozparsuje na jednotlivá pravidla a vrací pole s těmito pravidly.

DbCore

DbCore je název jmenného prostoru, který obsahuje třídy pro práci s databází v rámci jádra. Hlavní třídou je stejnojmenná DbCore, dědící z hlavní třídy jádra Core, a která vytváří instanci třídy DbPdo. DbCore pak obsahuje dvě metody, kdy každá z nich obsahuje připojovací procedury k jednotlivým databázím, klientské a hlavní.

DbPdo je pak třída rozšiřující klasickou PHP třídu PDO, která slouží k práci s databázemi. DbPdo pak tedy jednoduše přijme přihlašovací údaje k databázi a provede připojení.

Zbývající třídy DbWeb, DbPage a DbAction rozšiřují třídu DbCore a obsahují metody vztahující se ke svým jaderním třídám. Tedy DbWeb obsahuje metody, které využívá třída jádra Web, třída DbPage obsahuje metody, které využívá třída jádra Page a třída DbAction obsahuje metody, které využívá třída jádra Action.

5.2 Administrace

5.2.1 Architektura administrace systému

Co se týče architektury administrace, tak ta má podobné rysy jako architektura jádra systému, ovšem členění jednotlivých částí je jiné. Není zde kaskádové rozdělení na web, stránky a moduly, ale pouze na administrační část modulů a správu vzhledu.

Hlavní třídou administrace je CoreAdmin a ta, kromě základních tříd, vytváří instance tříd Admin, Authentication a v případě úspěšného přihlášení i User. Třída Admin, dědící z třídy CoreAdmin, pak dále vytváří instance tříd Layout a ModuleAdmin, což je administrátorská část konkrétního modulu. Tato část pak zpětně komunikuje s administrací prostřednictvím instance třídy AdminAction, podobně jako tomu bylo u jádra systému.

5.2.2 Popis tříd administrace systému

indexAdmin.php

Hlavní spouštěcí soubor u administrace se nazývá indexAdmin.php a jeho obsah je totožný s obsahem souboru index.php v jádru systému, až na jeden malý, ale podstatný, rozdíl. Zatímco z jádra je spouštěna třída Core ze stejnojmenného jmenného prostoru, tak u administrace je spouštěna třída CoreAdmin taktéž ze jmenného prostoru se stejným názvem, tedy CoreAdmin.

CoreAdmin

CoreAdmin je tedy hlavní třída celé administrace. Z počátku se vykonávají základní úkony, jako u jádra Core, a nemá tedy smyl jejich popis opakovat (viz kapitola 5.2 části Load, Lang, Smarty a Cache). Jsou zde pouze drobné odlišnosti, jako je nepřepisování URL při zaslání požadavku GET. Není pro to důvod, tzv. „hezká URL“ mají svůj význam hlavně při SEO optimalizaci a v administraci nejsou tedy potřebná. V souvislosti s tím zde není nutnost u výběru jazykové mutace brát v úvahu URL adresu.

Po těchto úvodních nastaveních probíhá autentizace a autorizace uživatele (viz Authentication). Při úspěšné autorizaci se pak povolí vstup do administrace a spustí se metoda, která spustí případné další metody dle zaslaného požadavku. V opačném případě se uživatel nepřihlásí a spustí se metoda načítající přihlašovací formulář. Nakonec se spustí metoda, která načte použité kaskádové styly a JavaScriptové kódy, sestaví z šablon a vygenerovaných dat výslednou stránku a tu zobrazí.

Authentication

Autentizace je proces, při kterém dochází k ověření identity daného subjektu (uživatele). V systému je toto ověřování vedeno pomocí uživatelského jména a hesla. Tyto

údaje jsou porovnávány s databází uživatelů, kde jsou uložena jejich jména a hesla převedená pomocí kryptovacího algoritmu SHA1 na otisk. Pokud tedy jméno a otisk hesla souhlasí, je vytvořen uživatel (třída User). V opačném případě vrací přihlašovací metoda hodnotu false. Kromě přihlašovací metody tato třída obsahuje metodu, na ověření zda je uživatel přihlášen a metodu na odhlášení.

User

User je třída, která obsahuje vlastnosti daného uživatele, tedy ID, jméno, roli a povolení k editaci modulů.

Admin

Tato třída vlastně slouží k tomu, aby rozhodla, jaké akce se budou v administraci vykonávat. Pokud není uživatel přihlášen spouští se metoda generující formulář. Pokud ale přihlášen je, načte se hlavní menu modulů a to tak, že se projde adresář s použitými moduly v klientské aplikaci, zjistí se, zda obsahují administrátorskou část a pokud ano, přidá se odkaz na tento modul do menu. Dále se zjistí, zdali již nebyl zvolen konkrétní modul a pokud byl, předá se řízení administrátorské části tomuto modulu. Dále se zjišťuje, jestli nebyla zvolena správa vzhledu, pokud ano předá se řízení třídě Layout. Nakonec se ověří zaslání POST požadavku a ověří se, zda nedošlo k odhlášení. Podle toho se pak zobrazí požadovaná stránka nebo přihlašovací formulář. Tato třída ještě navíc obsahuje dvě výše neuvedené metody. První zajišťuje stránkování seznamu, pokud jej administrační část modulu obsahuje, a druhá vytváří HTML kód pro výběr jazykové mutace administrace.

Layout

Layout je třída starající se o část administrace, která se věnuje správě vzhledu webové prezentace. Kromě této třídy existuje ještě třída ChangeLayout, která také spadá pod správu vzhledu, ale týká se pouze AJAXových požadavků a je popsána v následující kapitole. Třída Layout tedy obsahuje metody, vykonávající konkrétní činnosti jako přidání nové stránky, předgenerování obsahu a další. V následujícím textu tyto metody stručně popíši.

První dvě metody, o kterých se zmíním, slouží k obstarání funkce změny velikosti jednotlivých velikostí bloků stránky. Blok stránky je oblast, do které se vkládají jednotlivé moduly. Typicky může taková stránka obsahovat bloky jako hlavička, levý sloupec, střed, pravý sloupec a patička. První metoda tedy připraví stránku takovým způsobem, že se skryjí veškeré moduly a panely, a zůstane tak zobrazen pouze layout s ohraničením a textová pole pro zadání nových rozměrů. Po zadání rozměrů a odeslání provede metoda SetDivSize nové nastavení rozměrů a zpět se zobrazí kompletní administrace vzhledu.

Dále zde jsou obsaženy metody pro práci se stránkami. Jde hlavně o funkce, které by bylo zbytečné vykonávat pomocí AJAXu, jelikož by bylo stejně nutné překreslit celou stránku. Jde tedy o změnu stylu vzhledu stránky, vytvoření obsahu stránky, vytvoření nové stránky,

kterou lze vytvořit pomocí tří režimů. První režim umožňuje vytvoření stránky se stejným vzhledem jako je stránka aktuální, což se hodí, pokud je třeba vytvořit novou stránku, která bude vypadat stejně, ale bude mít jiný obsah. Při druhém režimu vlastně dochází ke zkopírování aktuální stránky i se svým obsahem, pouze název by měl být jiný. Tento režim se hodí v situaci, kdy je na stránce hodně modulů a u nově vytvářené stránky se bude měnit jen několik z nich, například jen hlavní modul. Poslední třetí režim umožňuje vytvoření zcela nové stránky a je tedy nutné zvolit nejen název, ale také základní rozvržení a styl vzhledu. Další metoda slouží k úpravě stránky, kdy lze změnit i základní rozvržení, ovšem v tom případě se ze stránky odstraní veškeré moduly. Poslední metodou pracující se stránkou jako takovou je její odstranění.

Další skupinou metod jsou metody pracující s moduly. Nalezneme zde například metodu, která vytvoří obsah daného modulu, tedy vygeneruje jeho vzhled pomocí jádra systému. Dále je možné vytvořit nový modul, takže tady nalezneme metodu, která vytvoří prázdný modul, tedy jakousi kostru k doplnění o požadovaný vzhled a funkce. Samozřejmě je tady i metoda k vymazání modulu. Pokud se modul vymaže, odstraní se zároveň ze všech stránek, které jej obsahovaly. Dále je zde možnost uložení informací o modulu, tedy názvu a popisu. Samotná editace modulu se pak provádí pomocí AJAXu a bude tedy popsána v kapitole věnující se této části.

Nakonec ještě zmíním pomocné metody jako je metoda, která vytvoří seznam stránek, které daná aplikace obsahuje. Dále metoda, která rozdělí velký seznam položek na několik menších. Tato metoda se používá k vytvoření stránkování při velkém počtu stránek nebo modulů. Dále je to metoda k vytvoření unikátního přepisovacího (rewrite) titulku stránky. Metoda, která vytvoří seznam šablon u daného modulu. Metoda vytvářející unikátní ID vloženého modulu. Metoda k vytvoření obsahu panelu modulů. Metoda sloužící k vytvoření seznamu stylů stránky. A nakonec to jsou malé metody, z nichž jedna dokáže získat z řetězce obsah umístěný mezi dvěma podřetězci. Další vyhledá v řetězci všechny zadané značky (tagy). A poslední v nalezeném tagu vyhledá požadovaný atribut.

AdminAction

Podobně jako byla u jádra třída Action, která se starala o propojení modulů s jádrem, tak v administraci k něčemu podobnému slouží metoda AdminAction. Zprostředkovává tedy administrační části každého modulu metody, pomocí kterých je možné vytvářet ovládací prvky administrace a další funkce. Obsahuje tedy metodu k vytvoření submenu (menu nižší úrovně zobrazené v horní části administrace, viz kapitola o návrhu vzhledu ovládacího rozhraní). Dále metody sloužící k vytvoření formulářových prvků jako jsou textová pole, tlačítka seznamy atd. Pak je to metoda, která vytváří tabulku se seznamem položek. U této tabulky jsou možnosti nastavení, zda mají jít položky editovat, vymazávat, přidávat či hromadně upravovat dle seznamu možných úprav. Zbývající metody jsou podobné jako v třídě Action, tedy získání obsahu požadavků GET a POST, zprostředkování připojení k databázi, získání seznamu jazyků včetně aktivně zvoleného a získání cest k adresářům modulu.

Form

Form je název jmenného prostoru a hlavní třídy, umožňující vytváření formulářových prvků v administraci. Tento jmenný prostor obsahuje dalších šest tříd reprezentující jednotlivé prvky. Všechny tyto třídy dědí z třídy Form a přebírají od ní vlastnosti název a jméno a metodu sloužící k vyrenderování. Dále jednotlivé třídy popíši. Třída CheckBox umožňuje vytvoření zaškrtávacího políčka (checkbox) a to jak v případě jednoho, tak i celé skupiny v závislosti na vstupních parametrech. Třída RadioButton umožňuje vytvoření tzv. přepínače puntíků (radio) a stejně jako CheckBox umožňuje vytvoření jednoho či celé skupiny. Lehce univerzální je pak třída Button, která umožňuje vytvořit nejen obyčejné tlačítko (button), ale i tlačítko k odeslání (submit), skryté tlačítko (hidden), obrázkové tlačítko (image) a tlačítko které slouží k výběru souboru z disku (file). Dále tady jsou třídy TextBox, která vytváří textové pole a třída TextArea, která vytváří textovou oblast. Poslední třída Select pak umožňuje vytvořit roletovou nabídku (select) s výběrem položek (option).

ModuleAdmin

ModuleAdmin je třída, která reprezentuje administrátorskou část konkrétního modulu. Je umístěna u každého modulu, který má vlastní administraci a nachází se v adresáři admin daného modulu. Musí být umístěna ve jmenném prostoru Modules a v podprostoru s názvem modulu (např.: \Modules\Guestbook) a implementovat rozhraní IAdminAction, které specifikuje povinné metody. První důležitou metodou je metoda, která připojí instanci třídy AdminAction pro přístup k administraci. Další je spouštěcí metoda, která se volá při zobrazení dané administrační části v administraci. A poslední třetí metoda se volá, pokud je zobrazen tento modul v administraci a zároveň byl zaslán nějaký POST požadavek. V této metodě se pak zařizuje obsloužení tohoto požadavku.

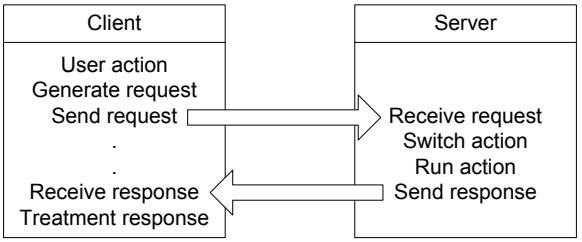
DbCoreAdmin

DbCoreAdmin je, podobně jako v jádru systému DbCore, název jmenného prostoru, v němž se nachází třídy pracující s databází. Jak už se stalo zvykem, hlavní třída má stejný název a rozšiřuje základní třídu administrace CoreAdmin. Podobně jako DbCore tak i DbCoreAdmin obsahuje dvě metody, které vytvářejí připojení k databázi klientské i hlavní. Opět je k tomu využito třídy DbPdo, která rozšiřuje klasickou třídu z PHP pro připojení k databázi PDO.

Další třídy v tomto jmenném prostoru pak obstarávají vlastní funkce a mají opět shodné názvy s třídami, které je využívají, jen mají přidán prefix „Db“. Jsou tady tedy třídy DbAdmin, DbAdminAction, DbLayout, DbLogin a DbUser. Všechny uvedené třídy pak rozšiřují hlavní třídu DbCoreAdmin.

5.3 AJAX administrace

Další část vytvářeného systému je implementována pomocí technologie AJAX. Ze své podstaty obsahuje serverovou část, která obstarává vykonání požadavků a vrácení výsledku, a klientskou část, která tyto požadavky generuje.



5.3 – Blokové schéma AJAXové komunikace

5.3.1 Serverová část

Serverová část tedy obsahuje PHP soubory a jejich stručný popis je uveden níže.

Ajax

Ajax je hlavní třída serverové části a stejně jako ostatní třídy je umístěna ve jmenném prostoru Ajax. Podobně jako třídy Core a AdminCore v jádru a administraci třída Ajax nejprve načítá základní nastavení, ovšem jen v omezené míře jelikož jich zde není tolik potřeba a zvláště záleží na rychlosti zpracování požadavku a vrácení výsledku. A dále zpracovává jazykové slovníky. Nakonec tato třída rozhoduje jaký „balík“ akcí se bude vykonávat. Jelikož je této technologie využito v části administrace se správou vzhledu, tak se načítá třída ChangeLayout, která se o tuto část stará.

ChangeLayout

Tato třída rozšiřuje hlavní třídu Ajax a provádí vlastní požadované úkony přicházející od klienta. Na několika dalších řádcích vysvětlím, k čemu všemu se používá.

Na začátku je připraveno připojení k databázi a přebrán požadavek POST, z něhož se zjistí, jaká funkce je požadována. Podle toho je vybrána příslušná metoda a ta je následně spuštěna. Třída tedy obsahuje práci se stránkami, u jejichž funkce není nutné celou stránku obnovovat. Je to tedy aktivování a deaktivování stránky, nastavení domovské stránky, načtení nových informací při změně jazyka stránky a uložení nového rozložení modulů na stránce.

Další metody se starají o správu modulů, případně zajišťují doplňkové funkce využité jinými metodami. Nejprve tady máme metodu, která je schopná načíst nový obsah modulu. Tato metoda jednoduše příjme název modulu a název stylu a vytvoří pro něj šablonu, kterou vygeneruje pomocí jádra a vrátí nový obsah. Další metoda v podobném duchu vrací nový

kaskádový styl. Při editaci zdrojového kódu modulu je samozřejmě nutné tento kód načíst. K tomu slouží další metoda, která dle zadaného modulu a souboru vrátí jeho obsah. Tato metoda spolupracuje s další třídou obstarávající základní práci s moduly a je popsána níže. V rámci editace modulů je také nutná základní podpora práce se soubory a adresáři. Existují zde tedy metody pro vytváření, mazání a přejmenovávání souborů a adresářů a samozřejmě zobrazení struktury daného adresáře. Speciální metoda pak řeší ukládání odeslaného zdrojového kódu. Dále je zde podpora pro nahrávání souborů, i když tato metoda přímo nevyužívá technologie AJAX. Nahrávání souborů na server je řešen pomocí vloženého neviditelného prvku „iframe" v němž samotné nahrávání probíhá. Po skončení nahrávání je pak vygenerován JavaScriptový kód, který uživatele informuje výsledku nahrání.

Podobně jako při práci se soubory lze i styly jednotlivých modulů vytvářet, přejmenovávat a vymazávat. Dále je zde implementována metoda pro změnu stylu modulu, změnu jazyka informací o modulu a vytvoření nového modulu. Zbývající metody jsou už pouze pomocné a umožňují vytvoření struktury adresáře a jeho zobrazení, načtení seznamu modulů a jejich stylů, promazání cache a získání obsahu z řetězce nacházejícího se mezi dalšími dvěma řetězci.

Module

Jak už bylo napsáno je tato třída pomocná a slouží k jednodušší práci s moduly. Obsahuje metody pro práci se všemi částmi modulu, tedy umožňuje spravovat PHP soubory, Smarty šablony, kaskádové styly, soubory JavaScriptu, administrační část a spravovat hlavní adresáře others, files a images.

DbCoreAjax

Jak již název napovídá, jedná se jako v předchozích případech o hlavní třídu sloužící pro připojení k databázi. Stejně jako předchozí hlavní databázové třídy i tato obsahuje dvě metody, pomocí kterých se může připojit k oběma dostupným databázím, tedy hlavní a klientské. Kromě této třídy AJAXová část obsahuje ještě třídu DbAjax, která se již stará o samotné databázové dotazy.

5.3.2 Klientská část

Klientskou část AJAXové technologie tvoří JavaScriptové soubory, které vykonávají funkce na straně klienta. Zde je jejich popis.

main.js

Jak již název napovídá, jedná se o hlavní soubor a obsahuje základní funkce. Je zde funkce, která usnadňuje zápis funkce pro vyhledání elementu podle ID. Dále funkce obnovující obsah zadaného elementu, tak že jej nejprve vymaže a pak zpětně vloží. Dále funkce, která vytváří objekt MouseEvent, který usnadňuje práci s myší a řeší rozdílnosti v implementaci

JavaScriptu napříč prohlížeči. Následující funkce také vytváří objekt, ovšem ten pouze vrací velikost aktuálně zobrazeného okna. Opět je zde z důvodu odlišnosti webových prohlížečů. Další univerzální funkce umožňuje hromadné označení všech zatrhávacích políček podle zadaného jména. Zbývající funkce pak již pracují s výsledkem vraceným od serveru a provádějí s ním následující akce. Nastaví vrácený výsledek jako obsah elementu (pomocí atributu innerHTML), nastaví vrácený výsledek jako hodnotu (value) daného elementu (používá se hlavně u formulářových políček), spustí vrácený výsledek jako JavaScriptový kód, přepíše obsah zobrazeného dokumentu vráceným výsledkem pomocí funkce write nebo zobrazí vyskakovací okno s výsledkem (použití hlavně při testování). Poslední funkce pak zobrazuje v informační oblasti animaci při načítání dat.

ajax.js

Tento soubor obsahuje jednu funkci Ajax, která umožňuje snadnou práci se zpracováním, přijmutím a odesláním požadavku na server. Tato funkce vytváří objekt, který obsahuje několik proměnných, pomocí nichž se specifikuje daný požadavek a jednu vnitřní funkci, která zajišťuje samotné připojení k serveru. Povinná proměnná je pouze jedna a ta definuje URL adresu, na kterou se má zaslat požadavek. Další proměnné umožňují definovat metodu přenosu, GET nebo POST, nebo připojit k požadavku data. Další dvě proměnné umožňují definovat tzv. „callback“ funkce. První definuje funkci, která se provede bezprostředně před odesláním požadavku. Je tedy možné uživatele informovat o načítání dat. A druhá specifikuje funkci, která se provede po úspěšném vrácení výsledku od serveru a přebírá dva parametry. První z nich je vrácený výsledek (XMLHttpRequest) a druhý parametr je specifikován v poslední proměnné funkce Ajax.

dragDrop.js

Tento soubor obsahuje funkce zajišťující přesouvání modulů pomocí techniky „chytni a pusť“, což umožňuje jednoduchou editaci obsahu každé stránky za pomoci myši. Tyto funkce jsou použity pouze pro přesouvání modulů, jelikož umožňují jejich přesné ukotvení v dané oblasti stránky. Pro přesouvání panelů (panel modulů, panel stránek atd.) jsou pak využity funkce umístěné v souboru layout.js, které umožňují plynulé přetahování těchto elementů po stránce.

Nejprve se spouští anonymní funkce, která je navázána na událost onload, které vzniká při dokončení načtení stránky prohlížečem. V této funkci probíhají počáteční nastavení, tedy určení elementů, které je možné přesouvat a které ne, a dále určení elementů, do kterých je možné přesouvat.

Dále jsou zde přítomny funkce, které jsou navázány na události myši jako jsou onMouseMove, což je událost, která vzniká při pohybu myši, dále onMouseDown, která vzniká při stisku tlačítka myši a onMouseUp, která vzniká při uvolnění tlačítka myši. Tyto funkce pak obstarávají samotný přesun elementů.

Nejprve se tedy čeká na vyvolání události onMouseDown. Navázaná funkce zjistí, zdali je kurzor myši nad elementem, který je nastaven jako „přetahovatelný“. Pokud je nad takovýmto elementem, tak se při pohybu myši a aktivování události onMouseMove příslušná navázaná metoda postará o přesun tohoto elementu. Přesněji řečeno se při každém pohybu myši zjišťuje, jestli je kurzor nad elementem, do kterého je možno „přetahovatelný“ element vložit, a pokud ano, tak se do tohoto elementu dočasně vloží.

Přetahování je dokončeno po uvolnění tlačítka myši a vyvolání události onMouseUp. Funkce navázaná na tuto událost má za úkol zjistit, kde bylo tlačítko myši uvolněno a podle toho se zachovat. Je zde několik možnosti, které mohou nastat. První možností je přetáhnutí elementu z panelu modulů do elementu stránky, sloužící ke vložení. V tomto případě nemůže dojít k přesunu, ale element se musí zkopírovat a vložit na dané místo. Další možností je přesun elementu z panelu modulů do prázdného místa, kamkoliv mimo určenou oblast pro vkládání. Pokud k tomuto dojde, nestane se nic a element modulu zůstane ve svém panelu. Třetí možností je přesun elementu mezi dvěma oblastmi stránky. Tady se vykoná klasický přesun, čili z původní oblasti se element vymaže a do nové se vloží. Čtvrtou možností je přesun elementu z oblasti stránky mimo oblast pro vkládání. V tom případě se daný element vymaže z původní oblasti. Nakonec zde máme ještě dílčí možnost, kdy je jeden „přetahovatelný“ element přesunut nad druhý. Pokud nastane tato situace, je jednoduše první element vložen na místo druhého a ten je odsunut níže pod první.

layout.js

V předchozím popisu souboru dragDrop.js, byly osvětleny funkce pro přesouvání modulů. Soubor layout.js, kromě jiného, obsahuje funkce pro přesun panelů. Nejedná se tedy o přesun z oblasti do oblasti, ale o plynulé přesouvání daného elementu po stránce, podobně jako přesun jakéhokoliv okna.

V tomto případě je vše řešeno trochu jinak, jelikož není nutné určovat, co a kam se může nebo nemůže přesouvat. Každý element (panel), který je určen k přesouvání, má definovánu funkci navázanou na událost onMouseDown (stisk tlačítka myši) tohoto elementu. Tato funkce zjistí a nastaví aktuální pozici elementu a připojí volání funkcí na základě událostí onMouseMove, onMouseUp a onSelectStart. Pokud je tedy element (panel) chycen a je vyvolána událost pohybem myši, příslušná funkce zjišťuje aktuální pozici kurzoru a nastavuje tak i pozici elementu. Po uvolnění tlačítka myši se pak tato i ostatní funkce odpojí a daný element tak zůstane na místě svého upuštění. Funkce SelectStart se pak stará pouze o to, aby při přetahování nebyl na pozadí označován text.

Další obsažené funkce se již samotným přetahováním objektů nezabývají. Je tady funkce, která obstarává změnu obsahu a vzhledu při změně stylu modulu. Dále funkce pro vytvoření unikátního ID. Funkce pro uložení nového rozložení stránky, resp. rozvržení modulů na stránce. Funkce pro deaktivaci odkazů na stránce. A nakonec funkce pro zjištění zda daný

element patří modulu nebo oblasti pro vložení a funkce vyhledávající element podle ID ve všech rodičovských elementech.

editBar.js

Kolekce funkcí v tomto souboru se stará o potřeby editace modulů. Jedna z nejdůležitějších funkcí tohoto úseku vytváří objekt reprezentující veškeré použité elementy v této části. Další velmi důležitou funkcí je funkce, která to vše odstartuje. Tato funkce zobrazí v dolní části stránky oblast, ve které se bude modul editovat a načte úvodní zobrazený soubor a strukturu adresáře. Jelikož jsou jednotlivé druhy souborů rozděleny do zvláštních záložek, tak jak bylo popsáno v části s návrhem, je v tomto souboru obsažena funkce na přepínání záložek. Dále jsou zde funkce starající se o jednotlivé druhy souborů, funkce načítající a ukládající obsah souborů. Menší funkce, které obstarávají adresářovou strukturu, tady její načtení, mazání, přidávání a přejmenovávání souborů a adresářů. Dále funkce pro práci se styly daného modulu, tedy jeho smazání, vytvoření nového či přejmenování. Samozřejmě nesmí chybět funkce pro uložení editovaného souboru a uzavření editační oblasti. K lepšímu uživatelskému komfortu zde pak jsou funkce pro maximalizaci editační oblasti, dále funkce, které v reálném čase aktualizují vzhled a obsah editovaného modulu či znovunačtení obsahu uloženého na serveru. Dále je zde část s nahráváním souborů, takže ani tato funkce nemůže chybět.

page.js

Jak název napovídá, jedná se o soubor obsahující funkci pro práci se stránkou. První metoda umožňuje skrýt panel se seznamem stránek, další dvě slouží k aktivování či deaktivování stránky. První provede samotné nastavení a druhá pak zařídí zobrazení příslušného tlačítka u stránky podle toho, zda je stránka aktivní nebo ne. Na stejném principu fungují i další dvě metody, které ovšem nastavují stránku jako domovskou. Následující funkce pak zobrazují či skrývají nabídky pro vytvoření nové stránky nebo její editaci. Poslední dvě funkce pak přepínají stránky v panelu stránek, pokud tento panel stránky obsahuje.

module.js

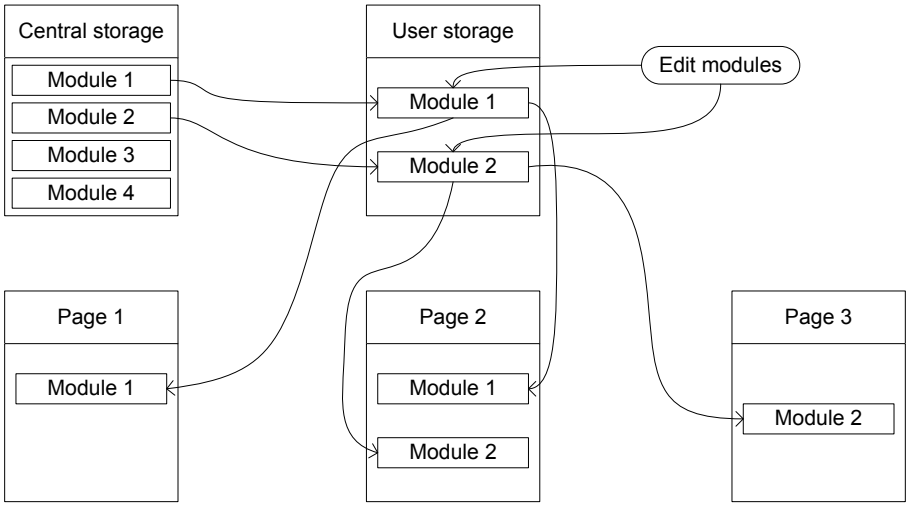
Poslední použitý JavaScriptový soubor je module.js, který obsahuje funkce pro práci s moduly, ale pouze tu část, která se netýká úprav modulu. Je zde tedy funkce pro zobrazení panelu k vytvoření nového modulu, dále funkce zobrazující panel s editací informací o modulu a funkce pro změnu jazyka v tomto panelu. Podobně jako v předchozím souboru jsou i zde stránkovací funkce tentokrát pro panel se seznamem modulů a také funkce pro skrytí tohoto panelu.

5.4 Moduly

Jelikož jsou moduly základním stavebním kamenem vytvořené webové prezentace, zaslouží si v tomto textu samostatnou část a konkrétnější popis.

5.4.1 Uživatelský pohled

Z pohledu uživatele je modul uzavřený objekt, který vykonává určitou činnost (zobrazuje fotogalerii, spravuje články atd.), má své umístění a vzhled. Tento objekt je možné umisťovat do daných oblastí na stránce webové prezentace. Na počátku tvorby prezentace jsou veškeré využitelné moduly uloženy v centrálním úložišti, kde jsou dostupné pro všechny klientské aplikace (webové prezentace). Při prvním použití modulu, kdy jej uživatel v administrátorské části přetáhne z panelu modulů na stránku, se modul z centrálního úložiště zkopíruje do úložiště modulů konkrétní webové prezentace. Takto je to řešeno hlavně z důvodu editace modulů, kdy není žádoucí, aby se úprava od jednoho uživatele projevila u všech klientských aplikací. Po této aktivaci si navíc modul zavede potřebná nastavení a je připraven k naplnění obsahu, pokud to tedy není jednoduchý statický modul bez citovatelného obsahu a bez administrační části. Úspěšně zavedený modul s vlastní administrací se pak objeví v hlavní administraci modulů v menu pod svým názvem. Tam je možné jeho obsah editovat.



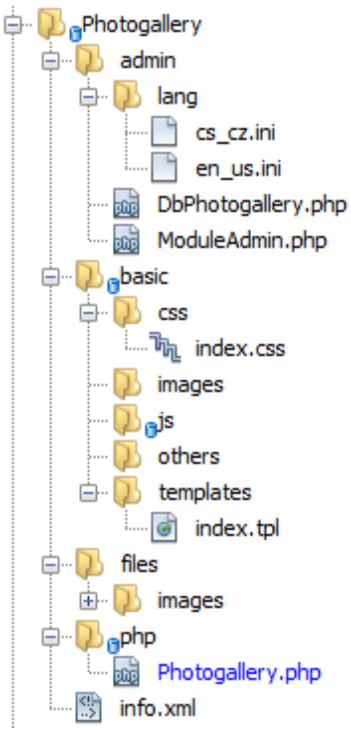
5.4 – systém modulů v aplikaci

V systému je dále podporována změna vzhledového stylu a funkčnosti daného modulu. Vše se odehrává v části administrace se správou vzhledu webové prezentace. Změna stylu je pak možná jednoduchou volbou z roletové nabídky v hlavičce každého modulu a ta se okamžitě projeví. Pokud se uživateli daný vzhled či funkčnost nezamlouvá má možnost v integrovaném editačním módu modul upravit. Upravovat lze pouze vložený modul na stránce a to z důvodu, aby byla zajištěna editace pouze modulu patřícího klientské aplikaci a ne modulu umístěného v centrálním úložišti. Druhý důvod je pak kvůli okamžitě viditelnému výsledku, kdy se při změně kódu automaticky upravuje vzhled modulu. Pokud má tedy uživatel příslušná oprávnění, může upravovat všechny části modulu, tedy kaskádové styly CSS, smarty šablony TPL, funkční kód PHP, administrační kód PHP, klientské skripty JavaScriptu a nahrávat soubory do adresářů files, images a others. Další možností je vytvoření zcela nového modulu. K tomu stačí stisknout

tlačítko pro přidání nového modulu, zadat název a potvrdit. Poté se vygeneruje prázdný modul, kde je vše potřebné připraveno. Dále se postupuje jako při jakékoli editaci jiného modulu.

5.4.2 Programátorský pohled

Z programátorského pohledu jsem v předchozích kapitolách naznačil, jak moduly vznikají a komunikují. Pro lepší přehled o funkčnosti modulů, zde uvedu příklad konkrétního modulu a tím je fotogalerie, na které popíšu vše důležité. Uvedená fotogalerie umožňuje zobrazení nahraných fotografií s názvem a popisem. Zobrazeny jsou vždy tři náhledy vedle sebe a až po rozkliknutí se zobrazí plná velikost. Administrační část tohoto modulu tedy umožňuje automatické vytvoření náhledu, vložení názvu a popisku a dále nastavení maximálního počtu zobrazených náhledů na stránku.



5.5 – adresářová struktura modulu

Každý modul má dánu předepsanou adresářovou strukturu. Hlavní adresář modulu se musí jmenovat stejně jako samotný modul, v našem případě je to Photogallery. Tento adresář pak obsahuje následující podadresáře. Podadresář php, ve kterém jsou umístěny PHP soubory zajišťující funkčnost modulu. Dále podadresář admin, který také obsahuje PHP soubory, ovšem ty zajišťují administrační část tohoto modulu. Do podadresáře files se pak umisťují soubory, které jsou dostupné všem stylům modulu. V našem případě adresář files obsahuje adresář images s nahranými fotografiemi. Zbývající podadresáře hlavního adresáře obsahují strukturu daného stylu. U fotogalerie je přítomen styl basic a tedy i stejnojmenný adresář, který dále obsahuje podadresáře css pro umístění kaskádových stylů, images pro uložení použité grafiky daného stylu, js pro JavaScriptové soubory, templates pro umístění Smarty šablon a others pro případné další soubory použité v konkrétním stylu. Hlavní adresář Photogallery pak ještě obsahuje soubor info.xml, který obsahuje název modulu a dále popis a titulek v několika jazycích, v našem případě je to angličtina a čeština.

Hlavní PHP soubor, který musí modul obsahovat, musí být pojmenován podle názvu modulu. V našem případě je to tedy soubor Photogallery.php. Tento soubor obsahuje stejnojmennou třídu, která je umístěna ve jmenném prostoru \Modules\Photogallery a implementuje rozhraní \Interfaces\IAction. Toto rozhraní definuje metodu SetAction, která jako parametr přijímá instanci třídy Action, která umožňuje přístup k jádru systému (viz předchozí kapitoly), a uloží ji do privátní proměnné action. Další povinná metoda definovaná rozhraním IAction je Init. Tato metoda za pomocí action získá připojení k databázi a nastaví ho do privátní proměnné connection. Následuje další povinná metoda CreateInit, která se spustí pouze jednou a to při prvním aktivování modulu a způsobí volání metody, která vytvoří potřebné tabulky v

databázi. Poslední metoda definovaná rozhraním je Run. Ta v našem případě zjistí aktuální zvolený jazyk, načte nastavení pomocí metody GetSettings, načte fotografie pomocí metody GetPhotos, připraví stránkování a výsledná data předá Smarty šabloně.

Smarty šablona (index.tpl) pak definuje základní HTML kód a s pomocí získaných dat vytvoří seznam náhledů fotografií. Fotografie jsou uspořádány po třech až do výsledné velikosti definované v nastavení maximálního počtu náhledů na stránku. Dále jsou zde definovány stránkovací šipky pro přesun mezi jednotlivými stránkami fotogalerie. O přepínání se pak stará samotný výkonný PHP kód. Vzhled celému modulu pak dodává kaskádový styl index.css.

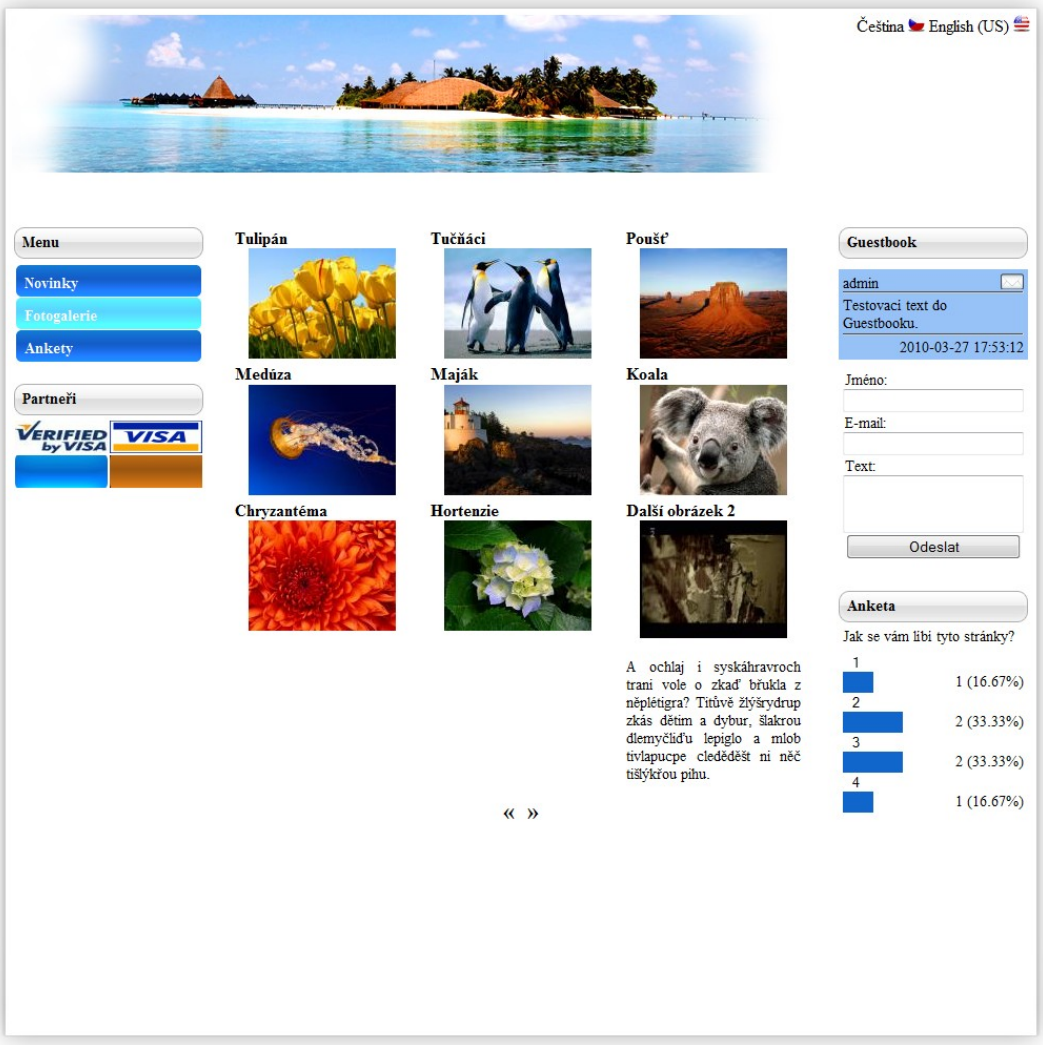
Tak to bychom měli základní část a teď je na řadě část administrační. Ta v našem případě neobsahuje jen základní třídu, ale také speciální třídu pro práci s databází a ve složce lang umístěné slovníky (anglický a český) v ini souboru. Nejprve trochu netradičně popíši třídu pracující s databází (DbPhotogallery). Ta je umístěna ve stejném jmenném prostoru jako ostatní třídy modulu a v konstruktoru přijímá vlastní připojení k databázi. Dále jsou zde umístěny metody, které vykonávají konkrétní požadované funkce na databázi. Je zde metoda, která načte veškeré uložené fotografie, resp. jejich názvy, s titulkem a popisem. Dále metoda, která načte pouze konkrétní fotografii s informacemi podle ID, nebo metoda, která získá název fotografie podle ID. Další důležité metody provádějí ukládání a mazání fotografií a ukládání a načtení nastavení.

Nyní přejdu k hlavnímu souboru administrace ModuleAdmin.php. Tento soubor obsahuje stejnojmennou třídu spadající pod jmenný prostor \Modules\Photogallery a implementující rozhraní \Interfaces\IAdminAction. Toto rozhraní definuje metodu SetAction, která přijímá jako parametr instanci třídy AdminAction a tu nastaví jako privátní proměnnou action. Díky této proměnné se načte připojení k databázi a vytvoří se instance třídy DbPhotogallery. Dále se v této metodě načte aktuálně zvolený jazyk a pomocí metody SetDictionary se načte konkrétní slovník. Nakonec se nastaví submenu na seznam a nastavení. Další povinnou metodou je Run. Ta se spouští při zvolení modulu aniž by byl odeslán POST požadavek. Zjistí se tedy, zdali byl zvolen seznam nebo nastavení a podle toho se spustí metoda GetList nebo GetSettings. Metoda GetList tedy získá seznam fotografií a pomocí metody CreateTable objektu action vytvoří tabulku, která je posléze zobrazena. Metoda GetSettings pak načte nastavení a zobrazí předvyplněný formulář s možností uložení nového nastavení.

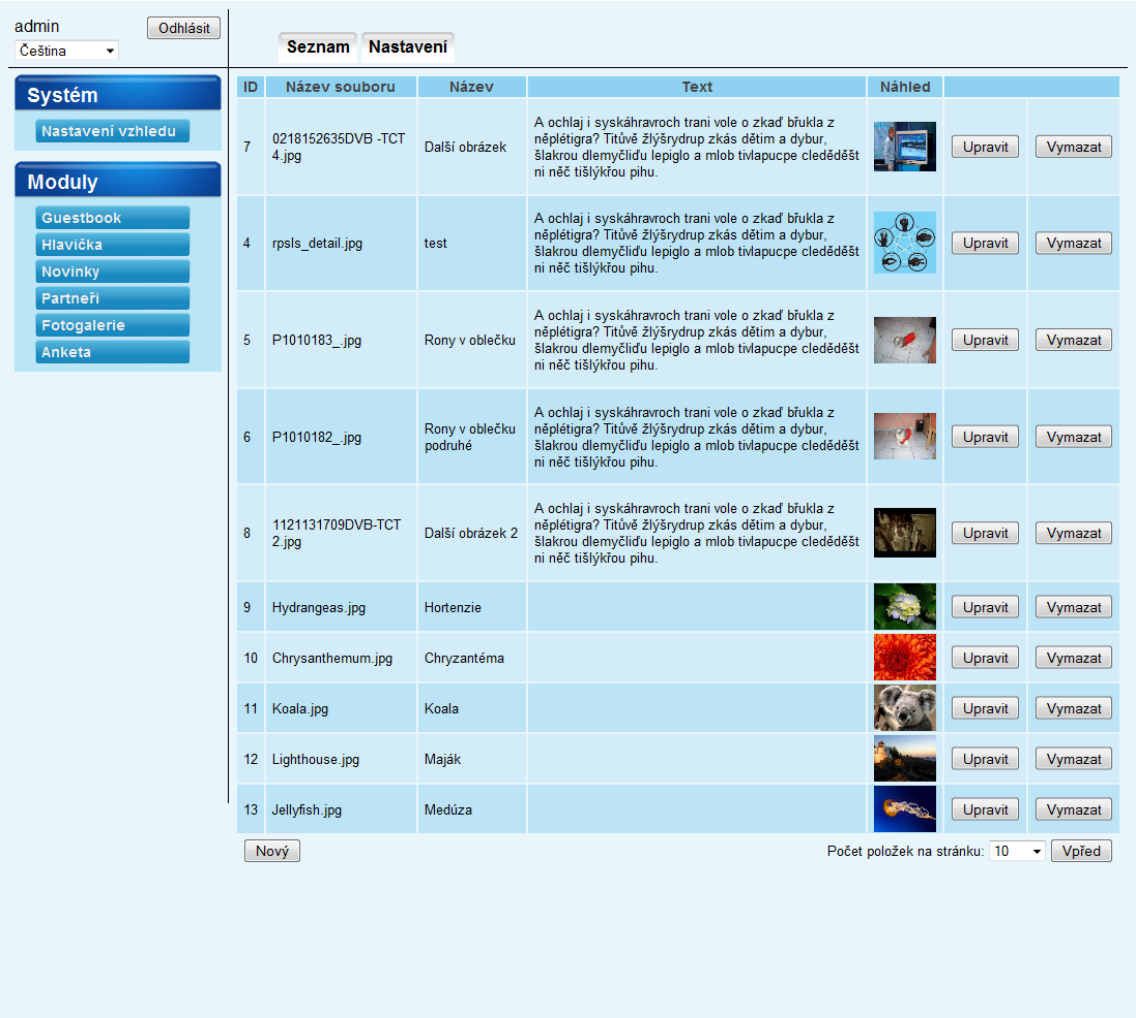
Poslední povinnou metodou definovanou v IAdminAction je DoPostBack, která se volá u vybraného modulu, pokud byl zároveň odeslán POST požadavek. V této metodě se nejprve rozhodne, k jaké akci má dojít, a podle toho se volají příslušné metody. V našem případě může dojít k následujícím akcím. Editaci, přidání či vymazání fotografie, dále uložení fotografie nebo nastavení a změně jazyka. Při volbě editace nebo přidání nové položky se volá stejná metoda Form a ta podle vstupního parametru určí, zda se jedná o editaci nebo přidání nové položky a případně zobrazený formulář předvyplní. Při uložení položky se volá metoda Save a ta si opět sama určí, zda se má vložit nová fotografie nebo editovat již stávající. Navíc se v této metodě

provádí vytvoření náhledu pomocí interní metody ImageResize. Další akce jsou již velmi jednoduché. Při mazání fotografie se tedy volá funkce Delete, která vymaže příslušný záznam z databáze a fyzicky odstraní fotografii i se svým náhledem z disku. Zbývá metoda SaveSettings, která jednoduše upraví stávající záznam s nastavením. Poslední akce je změna jazyka ve formuláři a ta pouze způsobí znovu načtení formuláře, pokud se jednalo o editaci.

6. Ukázky systému



6.1 – vygenerovaná klientská část prezentace



6.2 – administrace – editace obsahu modulu Fotogalerie

6.3 – Administrace – správa vzhledu

6.4 – Administrace – správa vzhledu s editací modulu

7. Závěr

Cílem mé diplomové práce bylo navrhnout a implementovat systém, díky kterému by bylo jednoduché vytvářet a spravovat webové prezentace. Systém by měl být pochopitelný pro běžné uživatele, ale zároveň je zde ponechána možnost úprav pro uživatele pokročilé, kteří si mohou pomocí HTML kódu a CSS stylů upravit vzhled modulů, nebo dokonce pro programátory, pro něž připravena podpora tvorby vlastních modulů čí úpravy stávajících.

Systém jako takový je plně připraven k použití, ovšem k jeho praktickému nasazení chybí větší počet modulů a kvalitních vzhledových šablon. Osobně jsem vytvořil několik ukázkových modulů, se kterými je možné vytvořit základní webovou prezentaci. Vytváření nových modulů ovšem není nijak složité a pustit se tak do něj může jakýkoli programátor znalý jazyka PHP. Při dalším vývoji by tak mohl vzniknout široce použitelný systém, který by nemusel sloužit pouze jako generátor webových prezentací, ale mohl by zpracovávat i složitější aplikace typu elektronického obchodu.

Literatura

1. **Croft, Jeff, Lloyd, Ian a Rubin, Dan.** *Mistrovství v CSS.* Brno : Computer Press, a.s., 2007. ISBN 978−80−251−1705−7.

2. **Holzner, Steven.** *Mistrovství v AJAXu.* Brno : Computer Press, a.s., 2007. ISBN 978−80−251−1850−4.

3. **Gilmore, Jason W.** *Velká kniha PHP a MySQL 5.* Brno : Zoner Press, 2007. ISBN 80−86815−53−6.

4. **Mrozek, Jakub.** *Smarty v PHP 5.* [Online] 1. 9 2006. http://smarty.ronnieweb.net/.

Přílohy

I. Kompletní zdrojové kódy aplikace (adresář „souce_code“)
II. Generovaná programátorská příručka v HTML formátu (adresář „documentation“)
III. Kompletní diagramy použité v analýze v HTML formátu (adresář „diagrams_complet“)